Institute of Information Technology,
University of Dhaka

IIT
University of Dhaka

Software Project Lab-II

Software Requirements Specifications & Analysis

for

Wallee

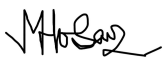
Personal Finance Manager

Prepared By

Mohammad Dardaul Hoque,
BSSE-1126

Proma Chowdhury
BSSE-1132

Supervisor’s Approval



09-03-22

Dr. B M Mainul Hossain

Table of Contents

Supervisor's Approval 1

1. Introduction 5
- Purpose 5
- Intended Audience 5
- Conclusion 6

2. Inception of Wallee 7
- Introduction 7
- Identifying the stakeholders 7
- Working with different viewpoints 7
  - An employed individual's point of view: 8
  - Developer's point of view: 8
- Conclusion 8

3. Elicitation 9
- Introduction 9
- Eliciting Requirements 9
- Collaborative Solution 9
- Quality Function Deployment 9
  - Normal Requirements 10
  - Expected Requirements 10
  - Exciting Requirements 10
- Acronyms and Definitions 11
- Usage Scenario 11
  - User Authentication 11
  - User-Profile Management 11
  - Transaction and Journal Management 12
  - Handling Journals 13
  - Selection of Currency Mode: 13
  - Summary Preview 14
  - Smart Receipt Reader 14
  - Wishlist 14

4. Scenario Based Modeling **15**
Introduction 15
Use-case Diagrams 15
Level 0: Wallee 16
Level 1: Wallee 17
Description of Level 1 : 19
Level 1.1: User Authentication 19
Level 1.2: User-Profile Management 21
Level 1.3: Transaction and Journal Management 23
Level 1.4: Selection of Currency Mode 25
Level 1.5: Summary Preview 26
Level 1.6: Smart Receipt Reader 27
Level 1.7: Wishlist 28
Activity Diagrams 29
ID 1: Wallee 29
ID 2: User Authentication 30
ID 3: User-Profile Management 31
ID 4.1: Transaction Management 32
ID 4.2: Journal Management 33
ID 5: Selection of Currency Mode 34
ID 6: Summary Preview 35
ID 7: Smart Receipt Reader 36
ID 8: WishList 37
Swimlane Diagrams 38
ID 1: Wallee 38
ID 2: User Authentication 39
ID 3: User-Profile Management 40
ID 4.1: Transaction Management 41
ID 4.2: Journal Management 42
ID 5: Selection of Currency Mode 43
ID 6: Summary Preview 44
ID 7: Smart Receipt Reader 45
ID 8: Wishlist 46

5. Data Based Modeling **47**
Introduction 47

Data Objects 47
Noun Identification 47
Final Data Objects 50
Cloud Firestore 51
Probable ER Diagram 52
Schema Diagram 52
Database Analysis 55

6. Class Based Modeling 56
Class based modeling concept 56
Identifying Classes 56
General Classification 56
Selection Criteria 59
List of verbs 60
Selected Classes: 62
Attributes and Methods Identification 62
Class Cards 66
CRC Diagram 70

7. Behavioral Modeling 71
List of Events 71
State Transition Diagrams: 74
ID 1 : User 74
ID 2 : FLP 75
ID 3 : ColorScheme 75
ID 4 : CurrencyMode 76
ID 5 : Journal 76
ID 6 : Transaction 77

8. Sequence Diagrams: 78

1. Introduction

This chapter is a part of our software requirement specification for the project "Wallee". In this chapter, we have focused on the intended audience for our project "Wallee".

1.1. Purpose

This document describes the Software Requirement Analysis of *Wallee - Personal Finance Manager*. It contains all the functional, non-functional, supporting requirements and expected requirements. At the same time it establishes a requirement's baseline for the development of the project. The requirements contained in the SRS are independent, uniquely numbered and organized by topics. But as time goes on, our SRS document is expected to evolve as users and developers work together to validate, clarify and expand its contents. The SRS serves as an official medium of expressing users' requirements to the developer and provides a common reference point for both the developer team and the stakeholder community.

1.2. Intended Audience

This document is intended for software developers, project managers, users, designers, testers.

• The work of the developer team that means the product which they have created can easily be verified by the customer whether it is acceptable to them or not by using this SRS document of Wallee.

• Project Managers will be able to plan milestones, delivery date and at the same time ensure whether the developing team is on track or not during the development of the application by using this SRS document.

• The designers will use this SRS as a basis for creating the application's design. The designers will continually refer back to this SRS to ensure that the system they are designing will fulfill the customer's needs.

• For developing the system's functionality, this SRS document will be used by the developers. The developers will link the requirements defined in this SRS to the software they create to ensure that they have created a software that will fulfill all of the customers' documented requirements.

• The testers will use this SRS in various phases of testing. They will use this SRS to derive test plans and test cases for each documented requirement. When portions of the software are complete, the testers will run their tests on that software to ensure that the software fulfills the requirements documented in this SRS. The testers will again run their tests on the entire system when it is complete and ensure that all requirements documented in this SRS have been fulfilled.

1.3. Conclusion

This overall document will help each and every person related to this project that includes users, project managers, designers, developers, testers, stakeholders to have a better idea about the project.

2. Inception of Wallee

2.1. Introduction

Automation is ubiquitous. Our aim behind the development of this project is to leverage automation in finance. We hope to present a smart and comprehensive solution to personal financial assessments.

Wallee is a Personal Finance Manager. It's a one-stop solution for users to monitor and predict their financial assets effortlessly.

2.2. Identifying the stakeholders

Stakeholders refers to any person or group who will be affected by the system directly or indirectly. Stakeholders include end-users who interact with the system and everyone else in an organization that might be affected by its installation.

Identification of the stakeholders were done from the information provided by the potential users. So far, we have managed to identify the following stakeholders of our project:

- Any employed individual
- Developers of Wallee

Stakeholders are expected to have basic computer literacy.

2.3. Working with different viewpoints

Different stakeholders expect different benefits from the system as every person has his own point of view. So, we have to recognize the requirements from multiple viewpoints. Different viewpoints of the stakeholders about the expected software are given below:

- An employed individual's point of view:
  - Helper tool for managing personal finance.
  - Receiving statistics about their financial status.
  - Smart monitoring of financial activities by the software.
  - Intuitive user interface.

- Developer's point of view:
  - Efficient use of computing resources by the system.
  - High uptime of database.
  - Responsive, non-blocking performance on client side.
  - Next to no possibility of runtime errors.

2.4. Conclusion

Our aim behind the development of this project is to leverage automation in finance. We hope to present a smart and comprehensive solution to personal financial assessments. We aim to design the software to provide high fidelity financial assessments with ease of use.

3. Elicitation

3.1. Introduction

Requirements Elicitation is a part of requirements engineering that is the practice of gathering requirements from the users, customers and other stakeholders. Many difficulties were faced, like understanding the problems,making questions for the stakeholders,limited communication with stakeholders due to a short amount of time and volatility. Though it is not easy to gather requirements within a very short time, these problems have been addressed in an organized and systematic manner.

3.2. Eliciting Requirements

Elicitation phase is mainly combining the elements for problem solving, elaboration, negotiation and specification. Without the collaboration of the stakeholder eliciting would have been really hard. We have finished the following tasks for eliciting requirements-

- Collaborative Solution
- Quality Function Deployment
- Usage Scenarios
- Elicitation work products

3.3. Collaborative Solution

Every stakeholder has their own requirements. There are some common and conflicting requirements of our stakeholders. That's why we followed the following steps to merge these requirements-

- Find the common and conflicting requirements.
- Categorize them.
- List the requirements based on stakeholder's priority points.
- Make final decisions about requirements.

3.4. Quality Function Deployment

QFD, Quality Function Deployment, is a technique that translates the needs of the customer into technical requirements for software. QFD mainly translates subjective quality criteria into objective ones that can

be quantified and measured and which can then be used to design and manufacture the product. It is a methodology concentrating on maximizing customer satisfaction from the software engineering process. So, we have followed this methodology to identify the requirements for the project. The requirements identified successfully by QFD are given below:

Normal Requirements

- Overview of current financial status
- Creation of account using appropriate credentials
- Basic journaling of transactions
- Periodic summaries of your financial status
- Manually add transactions
- Create Budgets
- Able to add preferred currency

Expected Requirements

- Responsive User Interface
- Ease of use
- Privacy & Security
- Being able to change login credentials
- Local Cache on device when disconnected from the internet
- User's data is readily available anywhere, anytime

Exciting Requirements

- SMS-Parsing using financial language processor
- Automatic addition of transactions
- Smart Receipt Reader

3.5. Acronyms and Definitions

3.5.1. **FLP -> Financial Language Processor :** This is a built-in module responsible for generating *Transactions* from plain text messages or shopping receipts.

3.5.2. **OCR -> Optical Character Recognition:** Used for generating raw text from paper receipts.

3.5.3. **SMS -> Short Message Service**

3.5.4. **CDN -> Content Delivery Network**

3.5.5. **POS -> Parts of Speech**

3.6. Usage Scenario

❖ User Authentication

➢ **Sign Up :** To access the features of the app, the user first needs to create an account. To create an account, the user needs to provide his phone number and for verification purposes, an OTP will be sent by Firebase Authentication services to the phone number. The user needs to provide the correct OTP to verify his phone number. To complete the signup process the user will have to provide his name.

➢ **Log in :** In case of logging in, the user needs to provide his phone number. To verify the phone number, an OTP will be sent by the Firebase Authentication Services to the user's phone number. The user can successfully log in by entering the correct OTP.

❖ User-Profile Management

➢ **Add Additional Info :** The user will be able to add additional infos like email address and profile photo if he wants to.

➢ **Specifying Base Currency :** The user will have to mandatorily add one base currency right after creating an account. The base

currency can only be specified in the very beginning and cannot be changed later. All user transaction related data will be stored in the base currency in the database.

- **Specifying Primary Balance :**
  The system will also store a User's *current balance.* Primarily the user has to provide a certain balance. After that, the net balance will be calculated on that balance by adding up all the *contributions* of their Associated Journals.

- **Change Phone Number:** The user will be able to change the phone number associated with the account anytime. In that case, first the user has to provide both his old phone number and new phone number and then an OTP will be sent to the new phone number and the user will have to complete the verification process by providing the correct OTP.

- **Add Phone Numbers for Targeted SMS Parsing :** The user will have to provide specific phone numbers or identifiers that they wish to monitor for transaction activity

❖ Transaction and Journal Management

Handling Transactions:

Transaction: *A Transaction is defined as an amount of change in the current contribution of a particular Journal. A transaction is -*

- *always associated with a particular Journal.*
- *Has a particular originator. That is, it can either be created by the user or proposed by the system itself.*
- *Has a particular time of entry*

Adding or Deleting Transaction: The user and the system both can add transaction entities. The system records transactions by parsing the incoming sms from phone numbers the user has specified for possible

transactions using FLP. The user needs to verify the system generated transactions. Apart from that, the user can delete any transaction anytime.

Handling Journals

Journal : A journal is a collection of Transactions executed in a particular category (e.g : Food, Rent, Salary). A journal is identified by its category. A journal will hold numerous information, such as:

- *A series of Transactions.*
- *Is either of type Income or Expense.*
- *Contribution: How much the journal adds / deducts from the user's net balance*
- *Net Transaction Volume: Total magnitude of transaction that a certain journal has accumulated so far.*
- *Frequency of access: How frequently a journal processes transactions.*
- *Timestamp of last access to journal.*

Add/Remove Journals: The user can create, edit and delete journals. The system can also automatically generate transactions - which are subject to the same treatment as just mentioned.

Search & Sort Journals: The user will be able to search through available journals, sort based on criteria (e.g. *access frequency*, *net volume of transactions*, *contribution*).

Modifying Journals: The system will propose a suitable journal category but the user will have the privilege to modify it.

❖ Selection of Currency Mode:

The user can change the currency mode to any other preferred currency anytime. Each currency mode will have an exchange rate. All the

transactions will be shown in the user's preferred currency. Note that, to add currency mode a base currency needs to be fixed first.

❖ Summary Preview

Specifying a time period :
An periodic summary of the transactions executed over a default period of time will be available to the user. The user will have to specify this period.

Specifying preview mode : The system can show the summary of journals in three different forms-

1. Default Tabular Form
2. Pie Chart
3. Bar Graph

Previews will be shown in any of the chosen forms by the user.

❖ Smart Receipt Reader

Users can scan paper receipts and be presented with auto-generated Transactions on their screen. They can then further verify which Transactions to keep. They can also modify them.

❖ Wishlist

- ➢ **Budget Creation**: The user can create budgets to fine tune expenditures. The budget name, time period, amount and preferred currency needs to be specified for each budget entity. Additionally, the user can turn on notifications to alert him when his expenditure is about to exceed the budget.

- ➢ **Goal Creation:** The user can set a goal of saving money in a particular field such as Education. He will have to specify the goal name, target amount, currently saved amount and desired date. The user can update the saved amount anytime.

4. Scenario Based Modeling

4.1. Introduction

Although the success of a computer-based system or product is measured in many ways, user satisfaction resides at the top of the list. If we understand how end users (and other actors) want to interact with a system, our software team will be better able to properly characterize requirements and build meaningful analysis and design models. Hence, requirements modeling begins with the creation of scenarios in the form of Use Cases, activity diagrams and swim lane diagrams.

4.2. Use-case Diagrams

A use case diagram of a software system describes the functionalities to be implemented by the software system. It is at a relatively higher level of abstraction compared to the other UML diagrams and will not include implementation details. Hence, it could be considered as a schematic view of the requirements for the software system. Together with the narratives for the use cases, the use case view provides a complete set of requirements for the system.

Primary Actors:

Primary actors are people, or at times even other systems, that require the assistance of the system under consideration to achieve their goal. They initiate the use cases of the system (business processes or application functionality). A use case within the system may have more than one primary actor, since more than one type of role may initiate the processes or functionality of the system.

Secondary Actors:

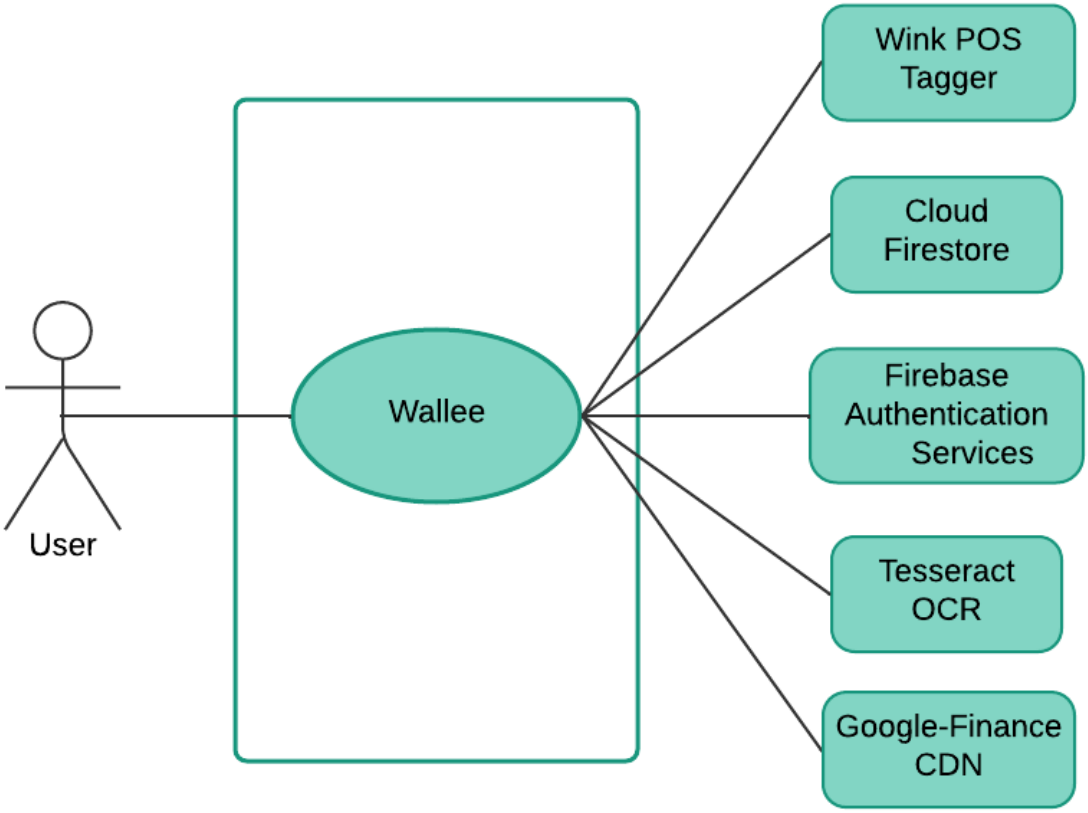
The system, and its use cases, often rely on other people, business processes, or applications in order to obtain a result or specific information required to achieve its goal. These are called Secondary Actors. A secondary actor is one from which the system requires assistance to complete the use case. A secondary actor never initiates the

use case. It is invoked by the system's use cases in order to obtain information or a result

Level 0: Wallee

Primary Actors: User

Secondary Actors: Tesseract Optical Character Recognition (OCR), Wink Parts of Speech Tagger, Cloud Firestore, Firebase Authentication Services, Google-Finance CDN



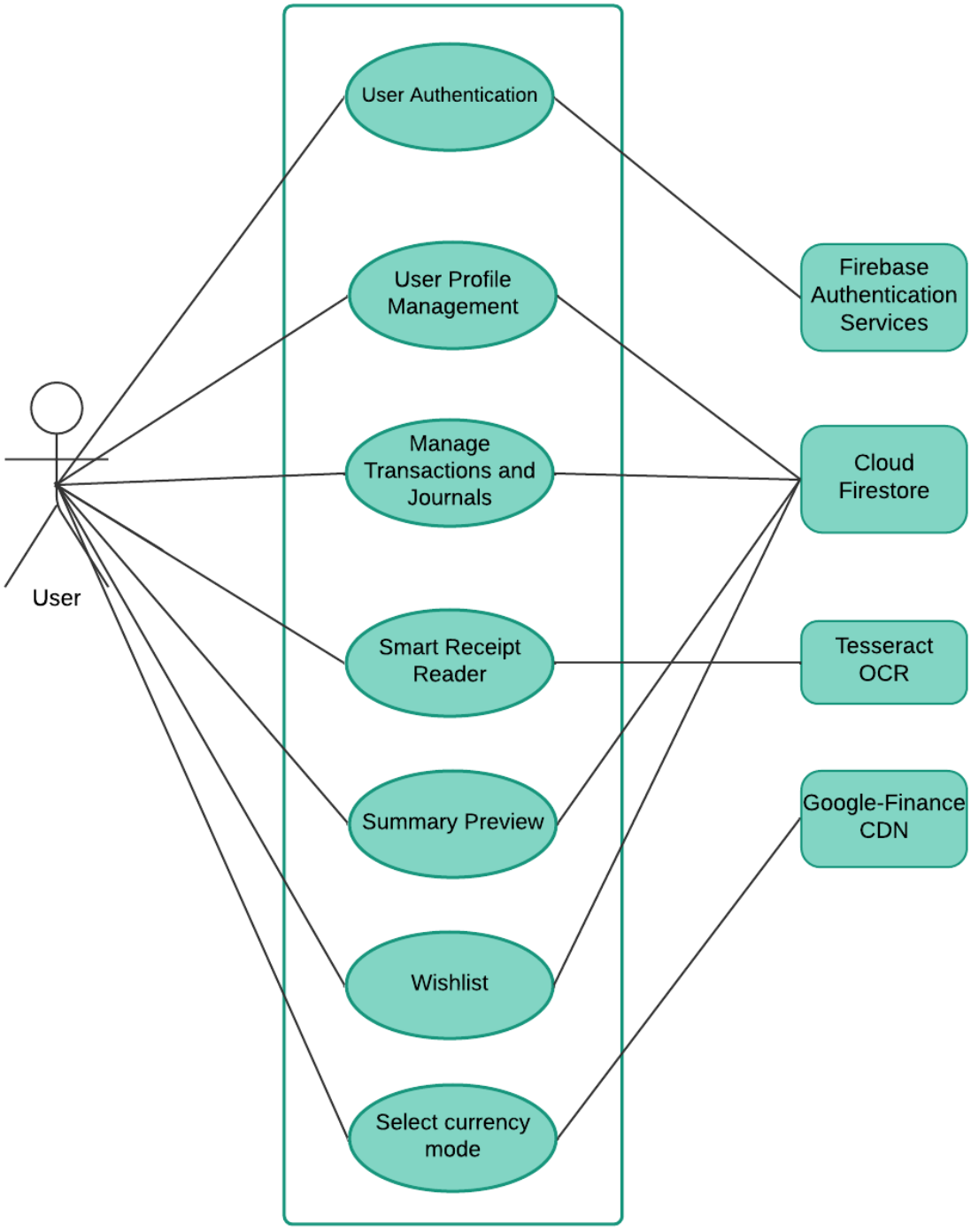
Description of Level 0:

This use case shows the low-level interaction between system and actors.

Level 1: Wallee

Primary Actor: User

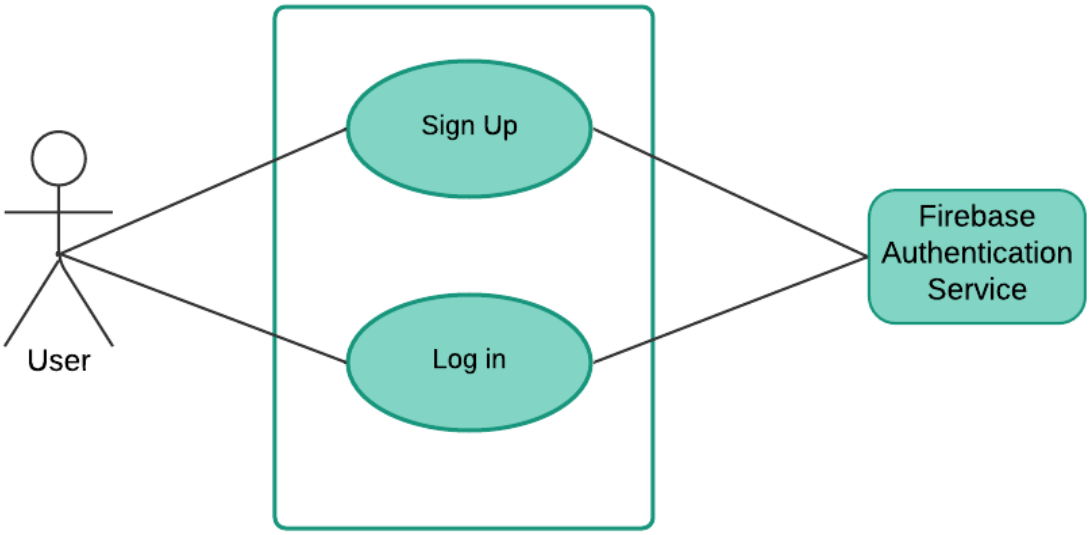
Secondary Actors: Tesseract Optical Character Recognition (OCR), Financial Language Processor (FLP),Cloud Firestore, Firebase Authentication Services, Google-Finance Content Delivery Network (CDN)


User Authentication
User Profile Management
Manage Transactions and Journals
Smart Receipt Reader
Summary Preview
Wishlist
Select currency mode
User
Firebase Authentication Services
Cloud Firestore
Tesseract OCR
Google-Finance CDN

Description of Level 1 :
From this level all the subsystems of the proposed main system and connectivity of those subsystems through actors has been explicit. From this level interaction between actors and subsystems will be clearer. Here, the whole system is divided into seven subsystems.

Level 1.1: User Authentication

Primary Actors: User
Secondary Actors: Firebase Authentication Service



Description of use case diagram level-1.1:

Sign Up : To access the features of the app, the user first needs to create an account. To create an account, the user needs to provide his phone number and for verification purposes, an OTP will be sent by Firebase Authentication services to the phone number. The user needs to provide the correct OTP to verify his phone number. To complete the signup process the user will have to provide his name.

Log in : In case of logging in, the user needs to provide his phone number. To verify the phone number, an OTP will be sent by the Firebase Authentication Services to the user's phone number. The user can successfully log in by entering the correct OTP.

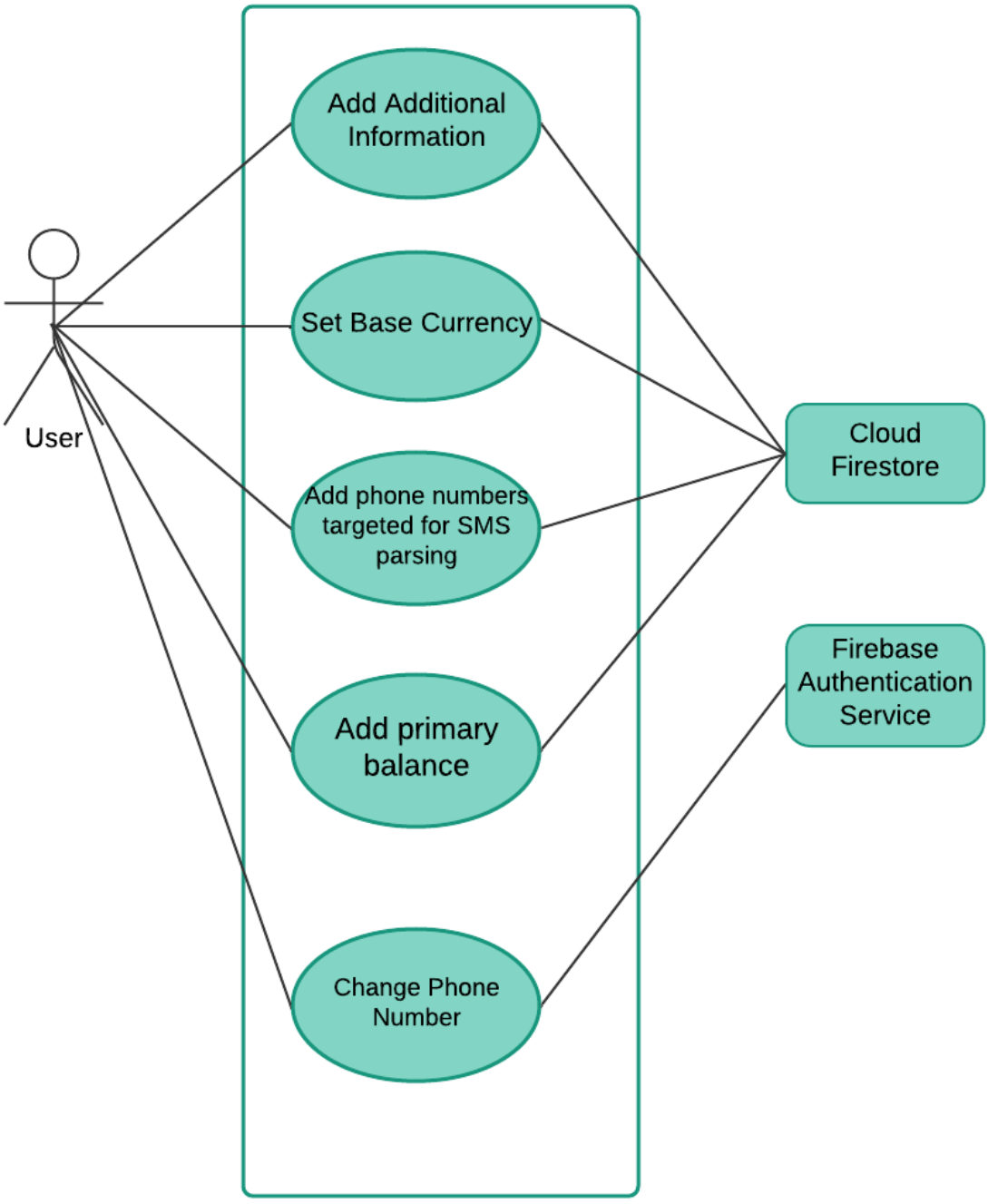
Action / Reply:

Action: The user provides Phone Number
Reply: Firebase Authentication will send an OTP to the phone number.

Action: The user enters the OTP
Reply: Firebase Authentication will check the validity of the OTP.

Level 1.2: User-Profile Management

Primary Actors: User
Secondary Actors: Firebase Authentication Service, Cloud Firestore



Description of use case diagram level-1.2:

1. **Add Additional Info :** The user will be able to add additional infos like email address and profile photo if he wants to.

2. **Set Base Currency :** The user will have to mandatorily add one base currency right after creating an account. The base currency can only be specified in the very beginning and cannot be changed later. All user transaction related data will be stored in the base currency in the database.

3. **Adding Phone Numbers for Targeted SMS Parsing :** The user will have to provide specific phone numbers or identifiers that they wish to monitor for transaction activity

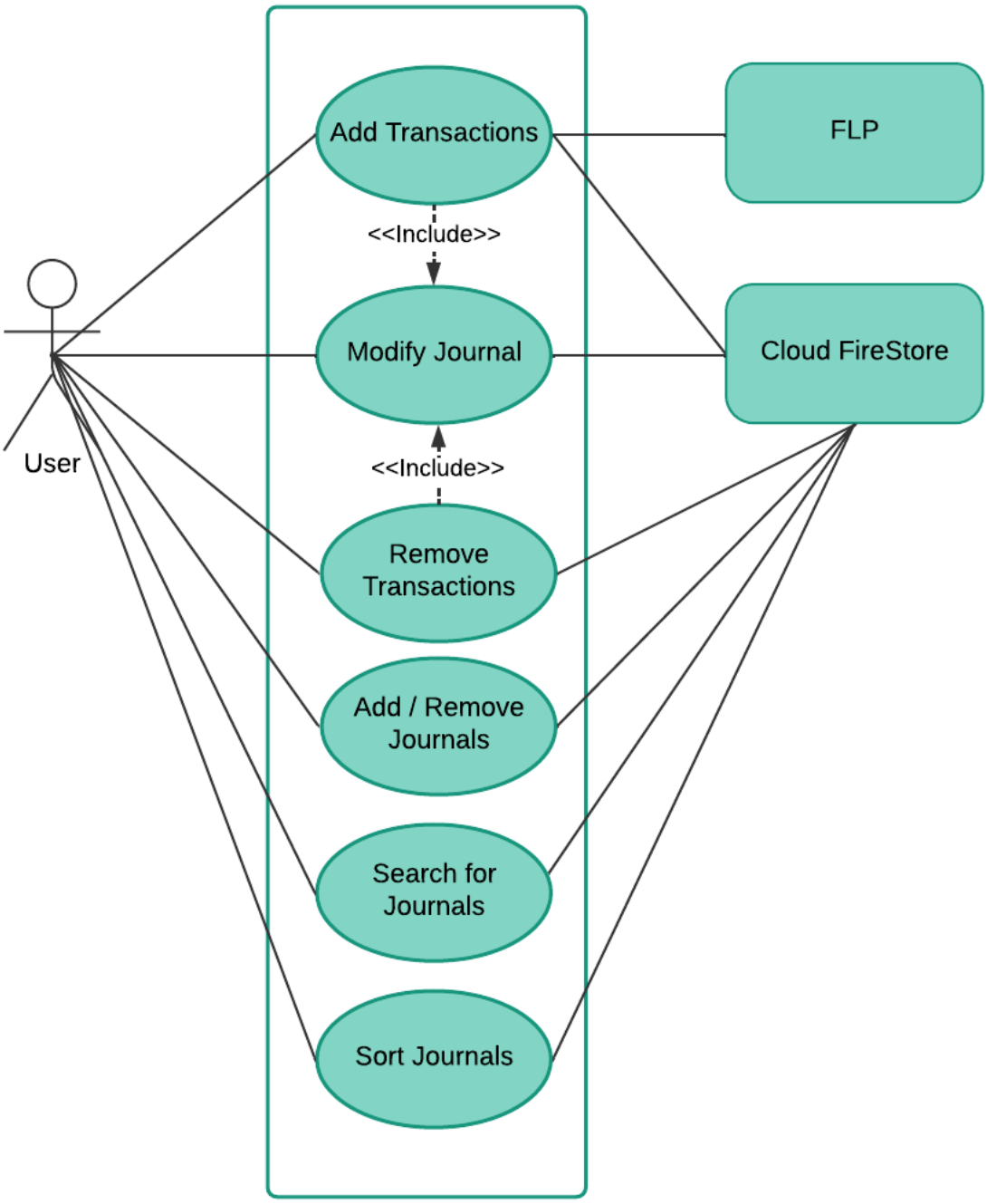
4. **Add Primary Balance :** The system will also store a User's *current balance.* Primarily the user has to provide a certain balance. After that, the net balance will be calculated on that balance by adding up all the *contributions* of their Associated Journals.

5. **Change Phone Number:** The user will be able to change the phone number associated with the account anytime. In that case, first the user has to provide both his old phone number and new phone number and then an OTP will be sent to the new phone number and the user will have to complete the verification process by providing the correct OTP.

Level 1.3: Transaction and Journal Management

Primary Actors: User

Secondary Actors: Financial Language Processor(FLP), Cloud Firestore



Description of use case diagram level-1.3

1. **Add Transaction:** The user and the system both can add transaction entities. The system records transactions by parsing the incoming sms from phone numbers the user has specified for possible transactions using FLP. Apart from that, the user can delete any transaction anytime.

2. **Modify Journals :** The system will propose a suitable journal category but the user will have the privilege to modify it.

3. **Remove Transactions :** The user can remove any transaction if he wants to.

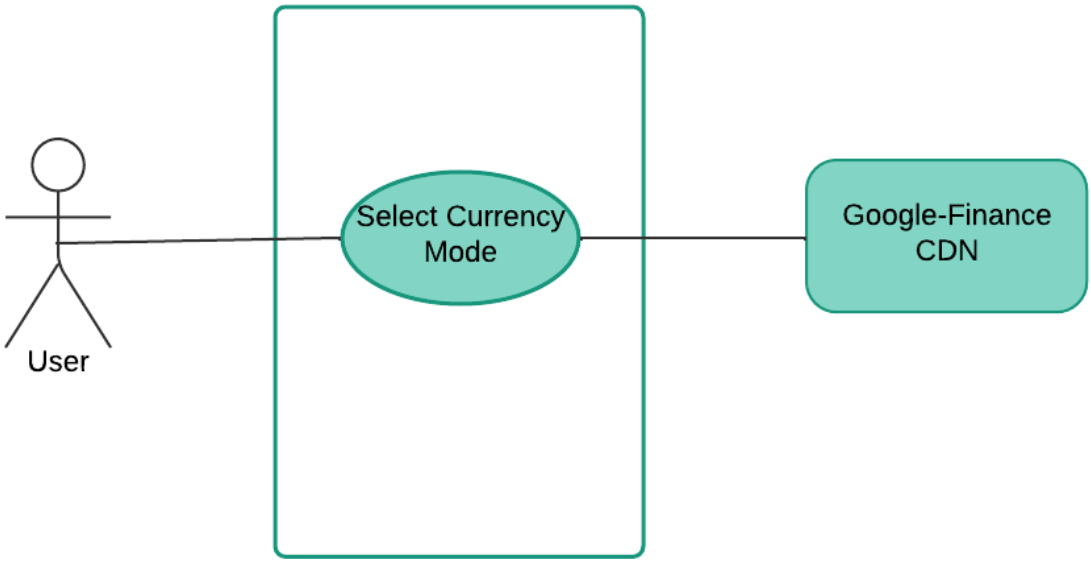
4. **Add/Remove Journals :** The user can create journal entries and add transactions to it. They can also delete any journal if they want.

5. **Search for Journals:** The user will be able to search through available journals.

6. **Sort Journals :** The user will be able to sort the journals based on criteria (e.g. *access frequency, net volume of transactions, contribution*).

Level 1.4: Selection of Currency Mode

Primary Actors: User
Secondary Actors: Google-Finance CDN

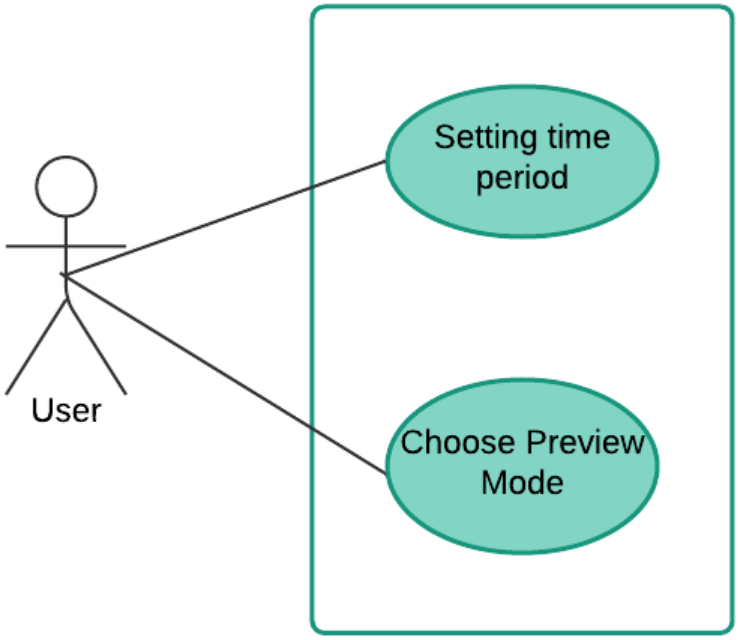


Description of use case diagram level-1.4:

Select Currency Mode : The user can change the currency mode to any other preferred currency anytime. Each currency mode will have an exchange rate provided by the Google-Finance API.

Level 1.5: Summary Preview

Primary Actors: User
Secondary Actors:

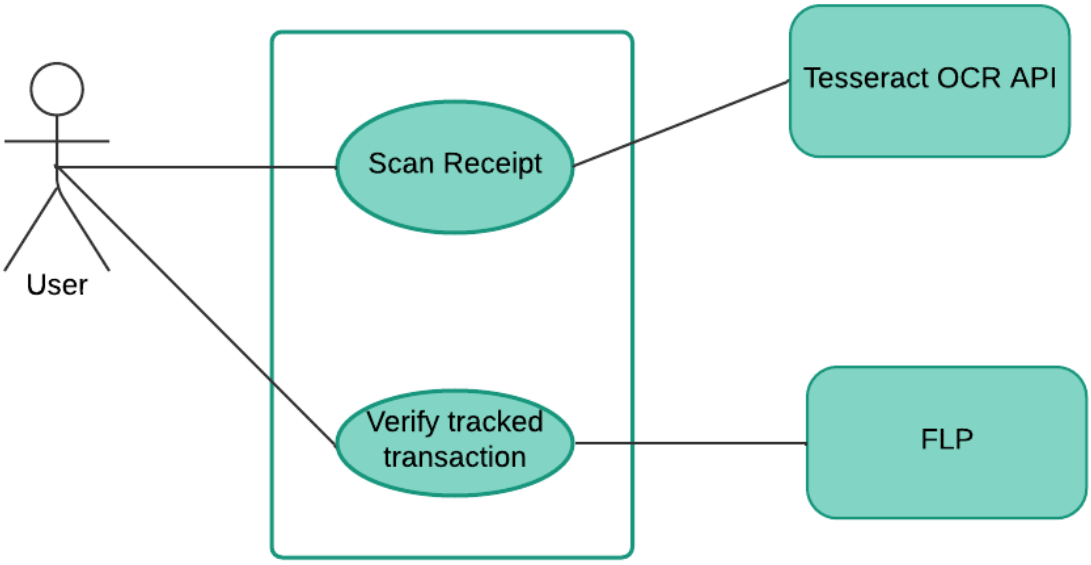


Description of use case diagram level-1.5:

1. **Setting the period :** An periodic summary of the transactions executed over a default period of time will be available to the user. The user will have to specify this period.
2. **Choose Preview Mode :** The user can set the preview mode e.g bar graph, pie chart, tabular format of the summary of journals.

Level 1.6: Smart Receipt Reader

Primary Actors: User
Secondary Actors: Tesseract OCR, Financial Language Processor

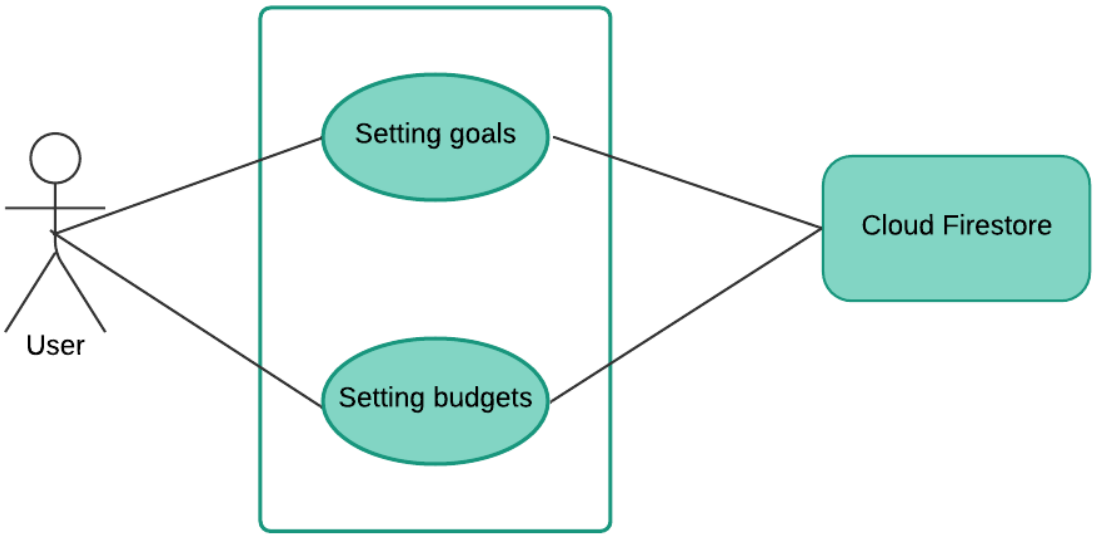


Description of use case diagram level-1.6:
Scan Receipts : The user will scan receipts and Tesseract will recognize the text within the digital image.

Verify Tracked Transactions : The user can verify the system proposed transactions tracked through parsing the receipt texts using FLP.

Level 1.7: Wishlist

Primary Actors: User
Secondary Actors: Cloud FIrestore



Description of use case diagram level-1.8:

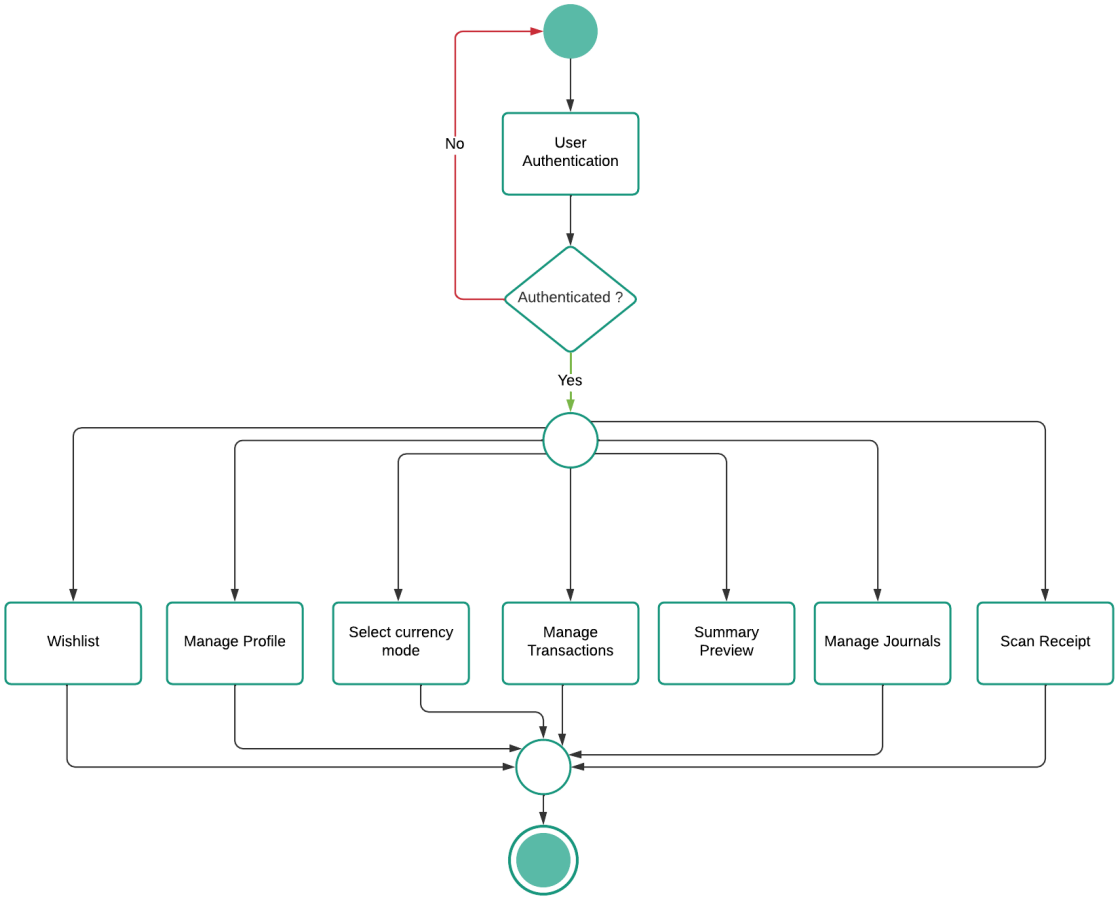
1. **Setting Budgets :** The user can create budgets to fine tune transactions. Additionally, the user will have to specify the name, time period, amount and preferred currency for each budget entity. The user can turn on notifications to alert him when his expenditure is about to exceed the budget.

2. **Setting Goals :** The user can set a goal of saving money in a particular field such as Education. He will have to specify the goal name, target amount, currently saved amount and desired date. The user can update the saved amount anytime.

4.3. Activity Diagrams

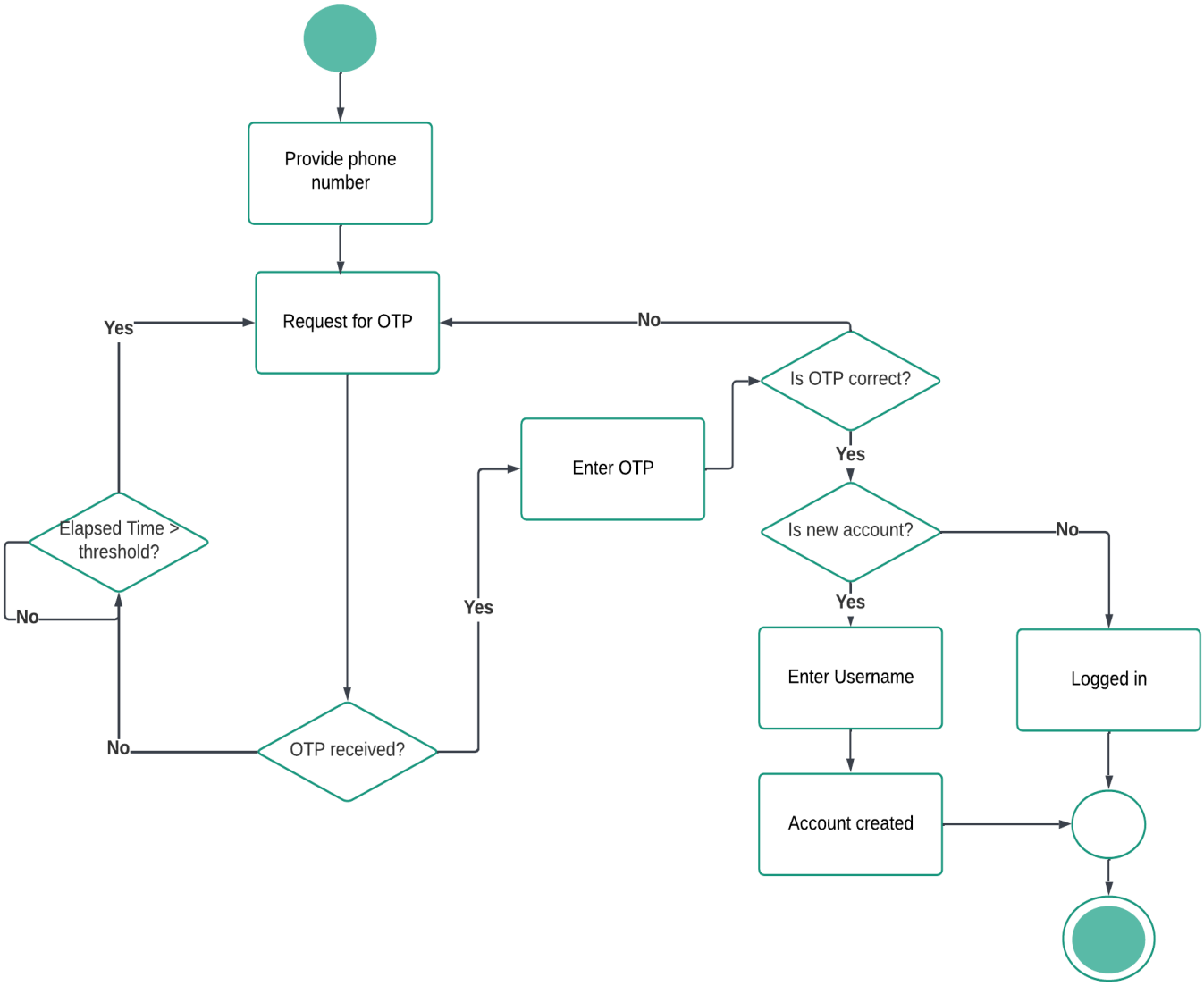
ID 1: Wallee

Reference: Use-Case Diagram 1



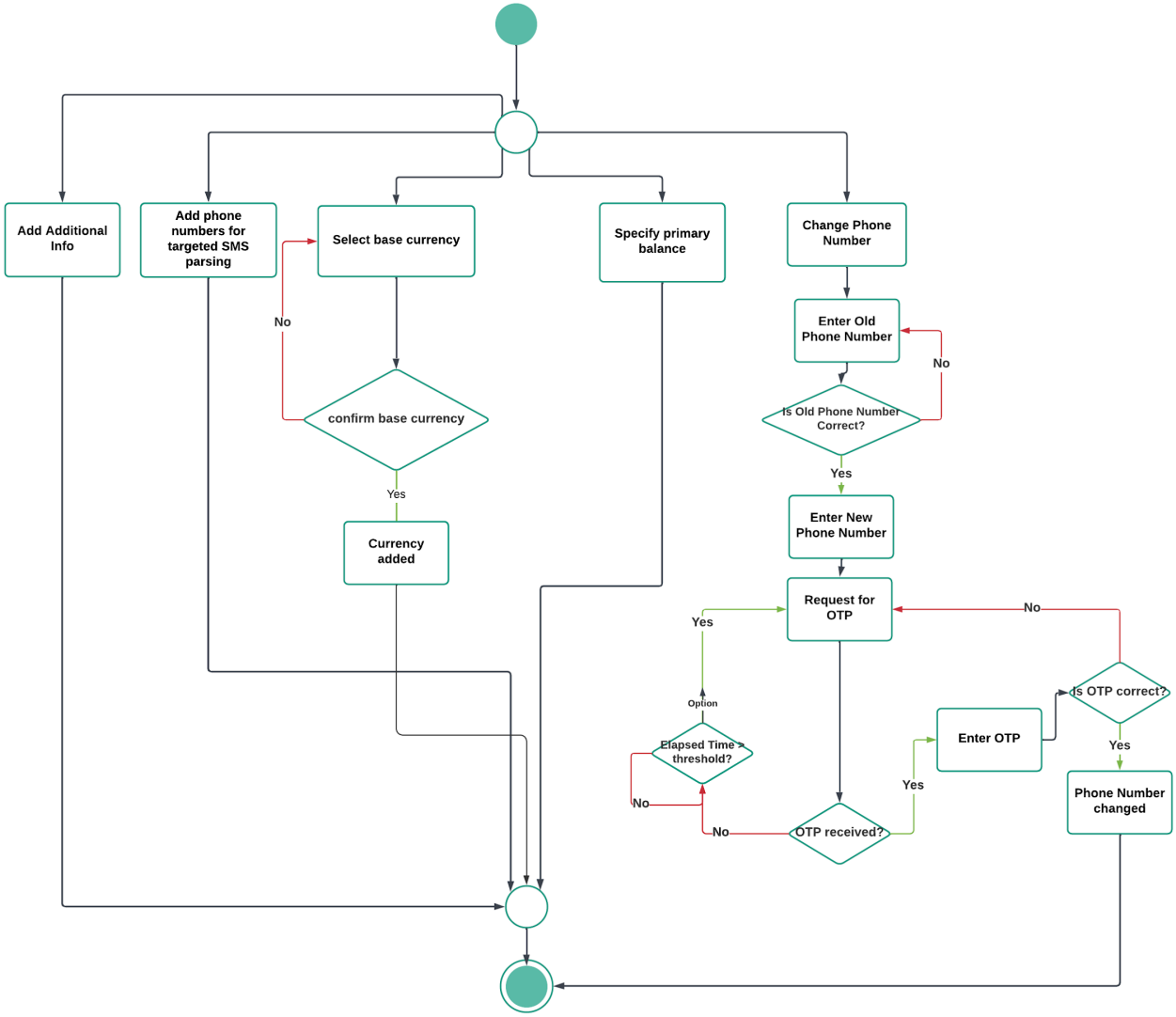
ID 2: User Authentication

Reference : Use-Case Diagram 1.1



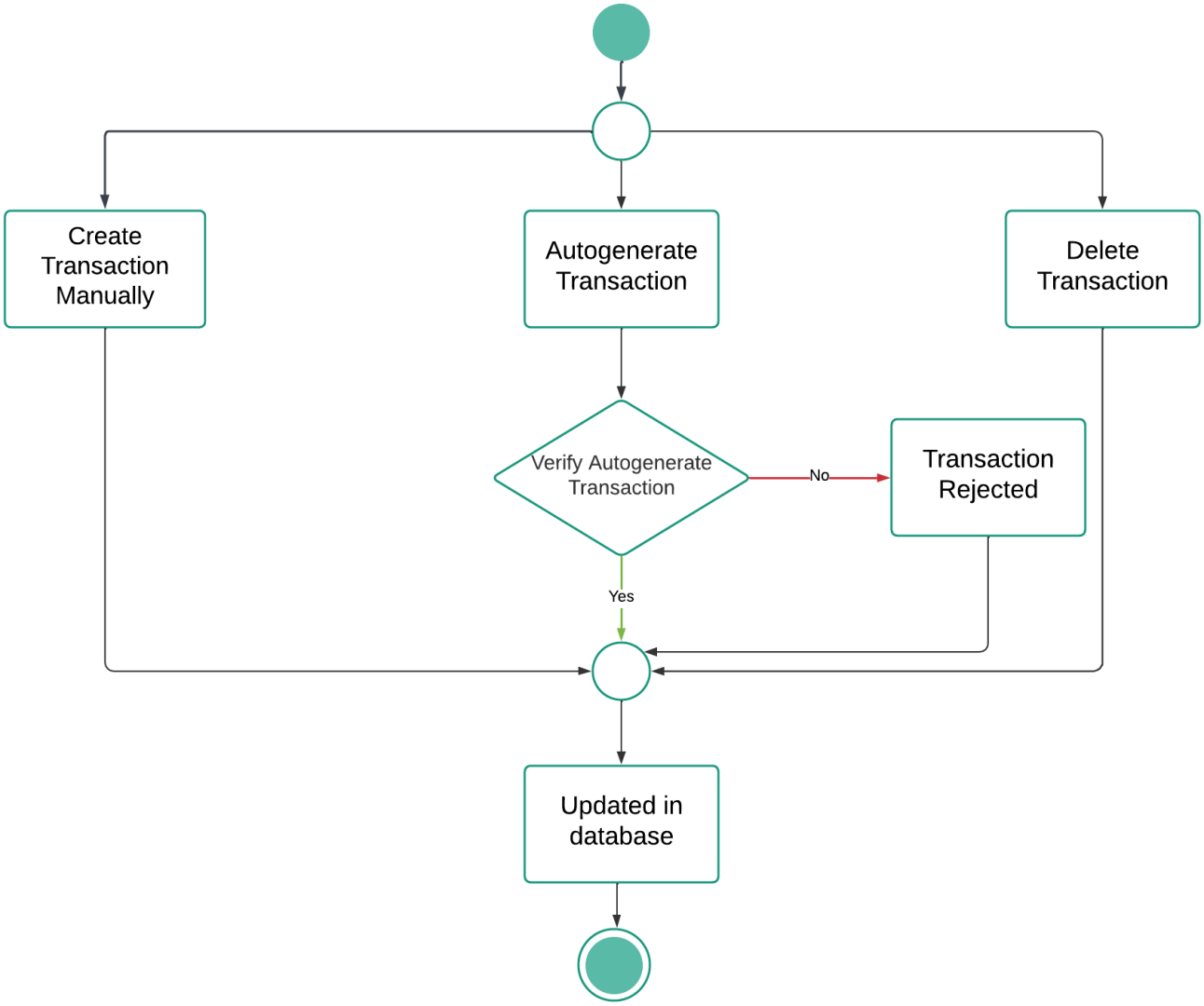
ID 3: User-Profile Management

Reference: Use-Case Diagram 1.2



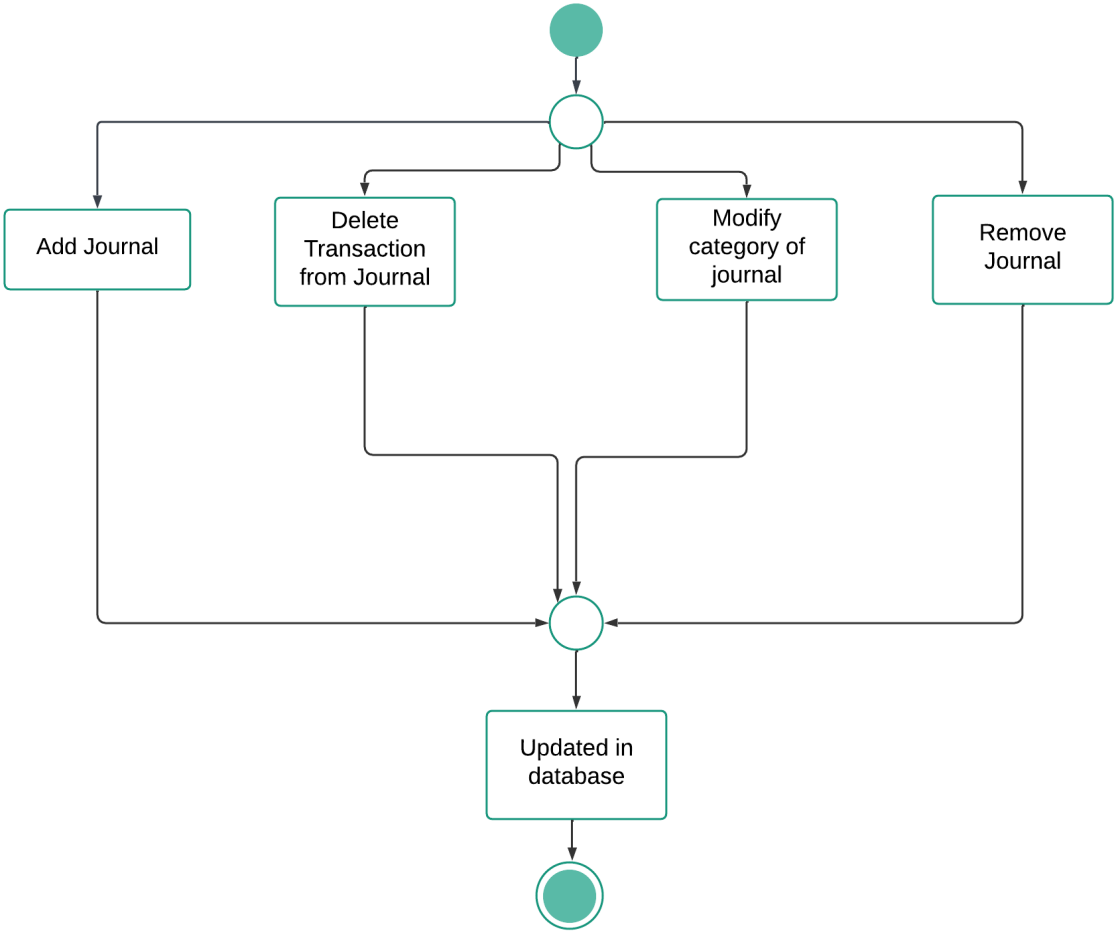
ID 4.1: Transaction Management

Reference: Use-Case Diagram 1.3



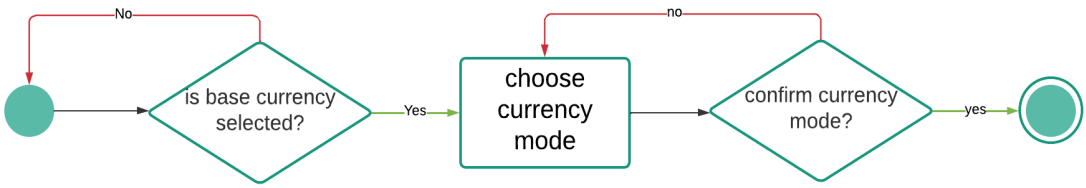
ID 4.2: Journal Management

Reference: Use-Case Diagram 1.3



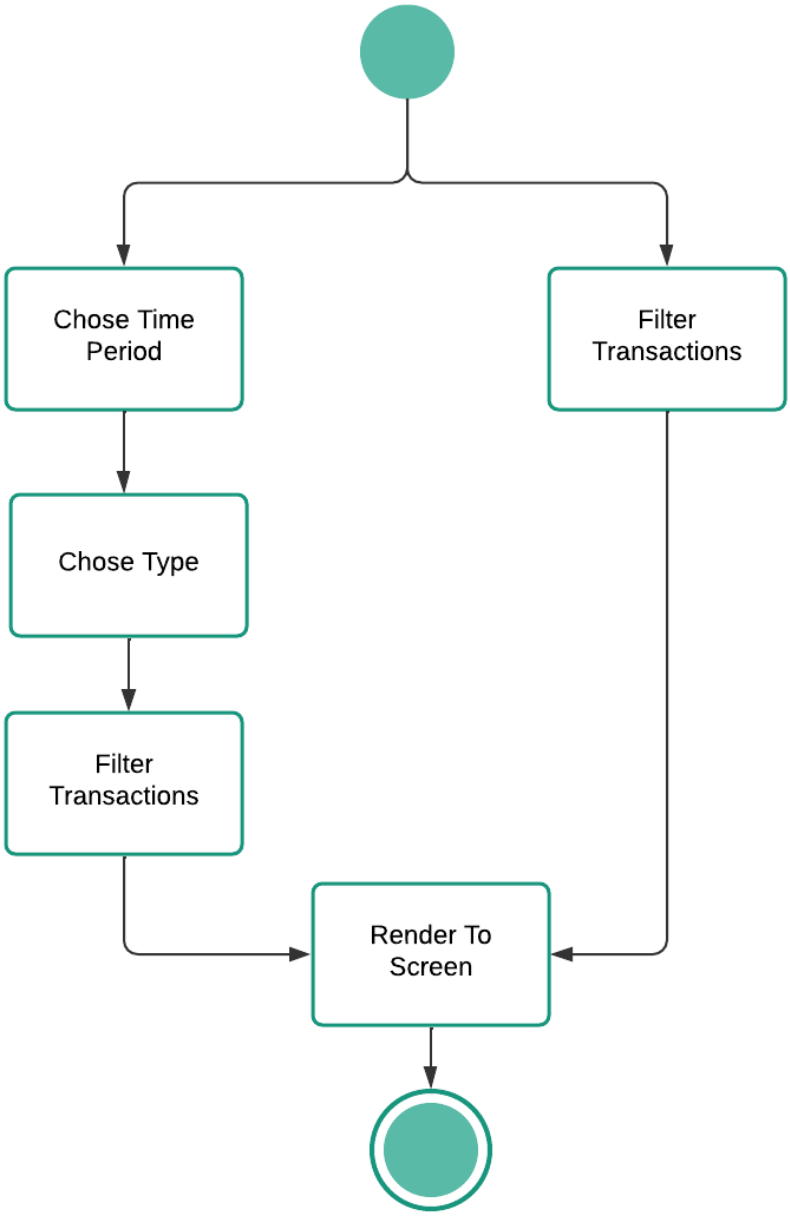
ID 5: Selection of Currency Mode

Reference: Use-Case Diagram 1.4



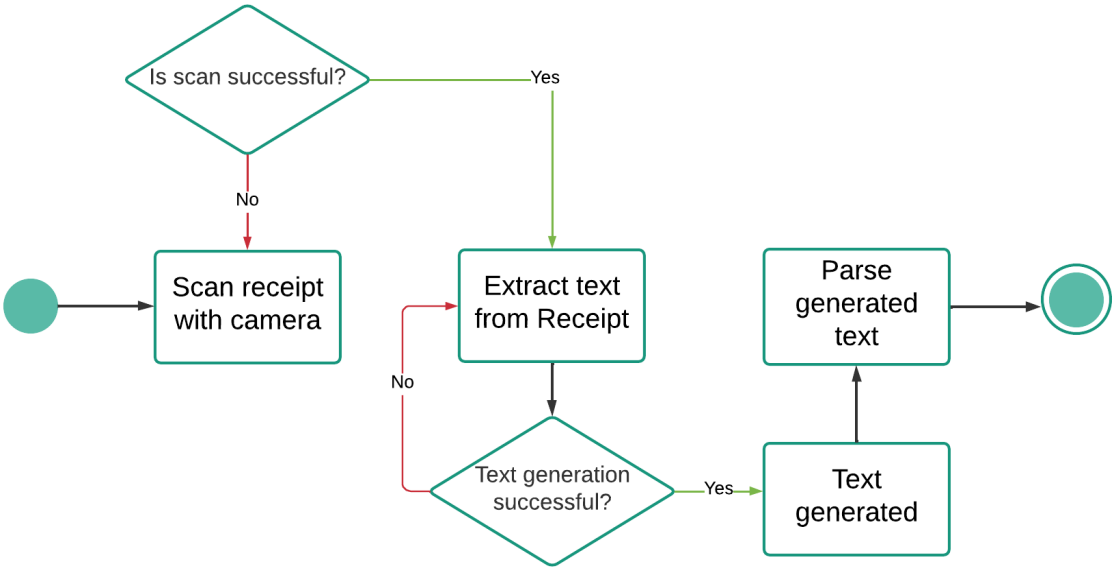
ID 6: Summary Preview

Reference: Use-Case Diagram 1.5



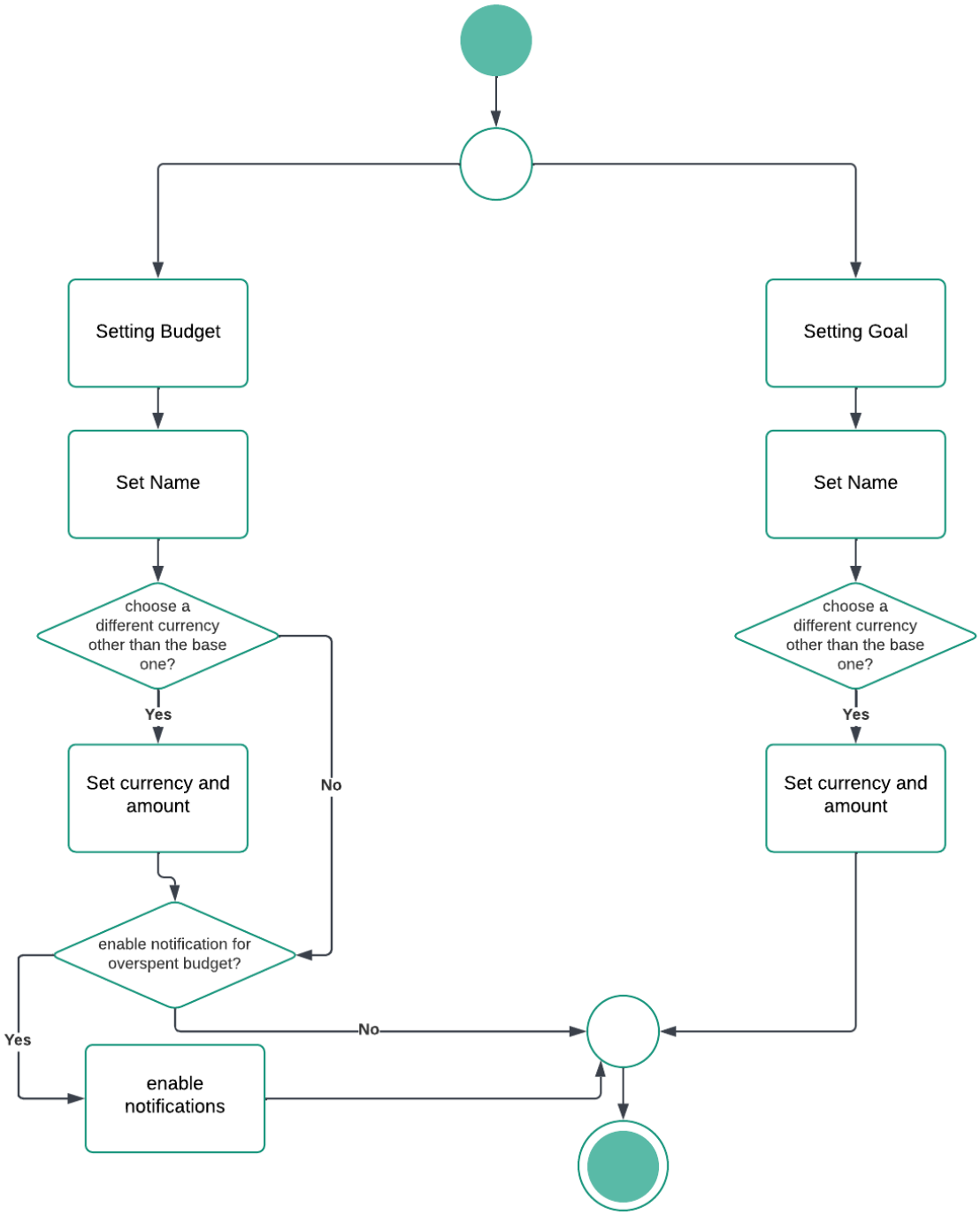
ID 7: Smart Receipt Reader

Reference: Use-Case Diagram 1.6



ID 8: WishList

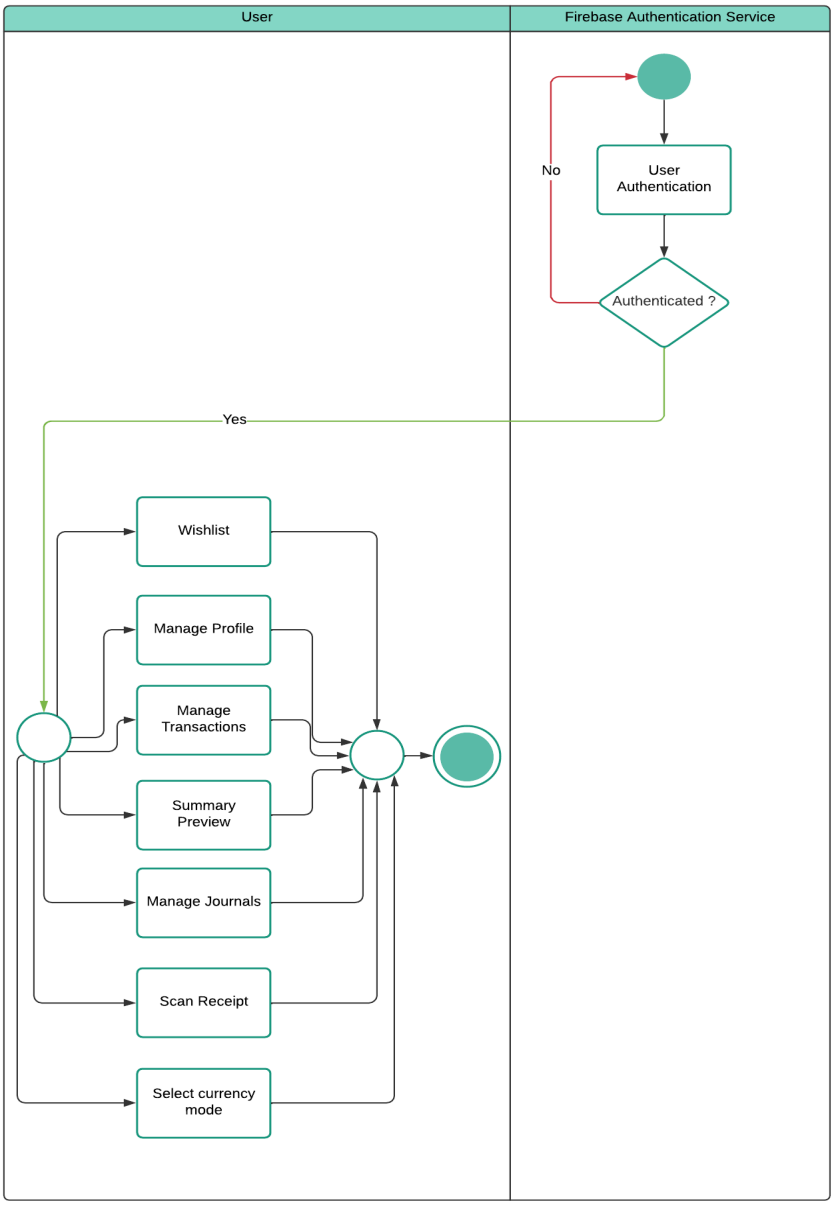
Reference: Use-Case Diagram 1.7



4.4. Swimlane Diagrams

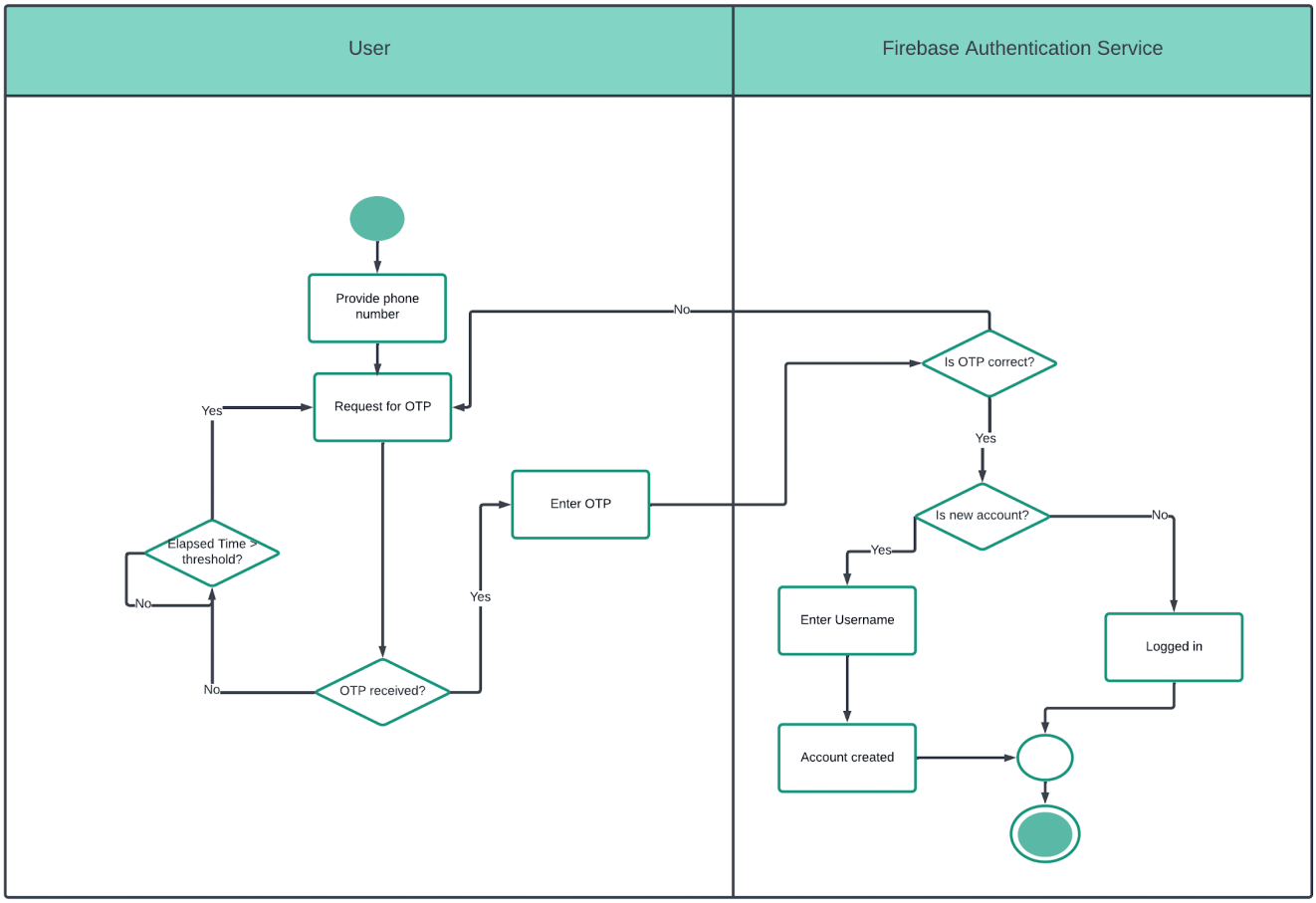
ID 1: Wallee

Reference: Use-Case Diagram 1



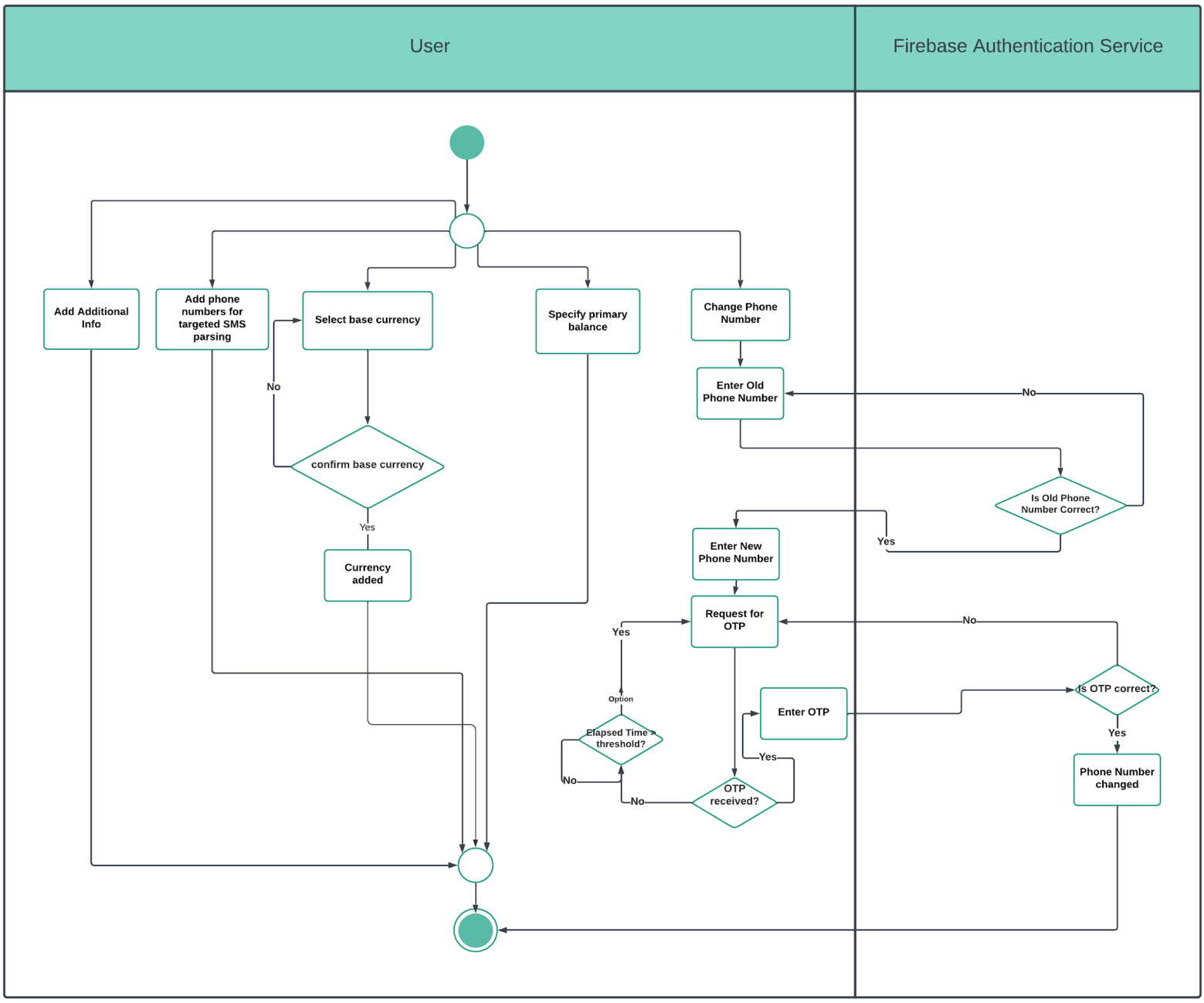
ID 2: User Authentication

Reference: Use-Case Diagram 1.1



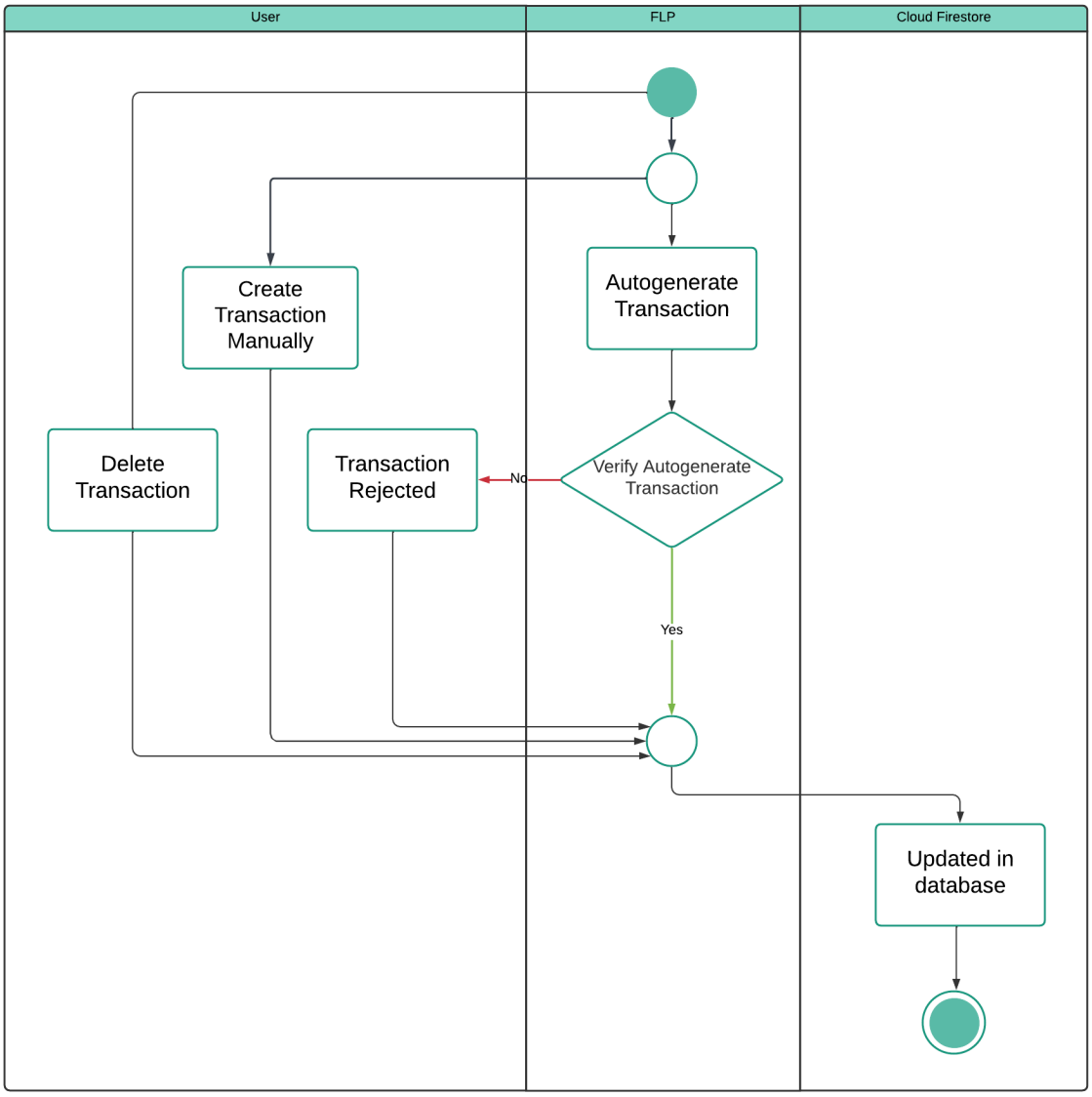
ID 3: User-Profile Management

Reference: Use-Case Diagram 1.2



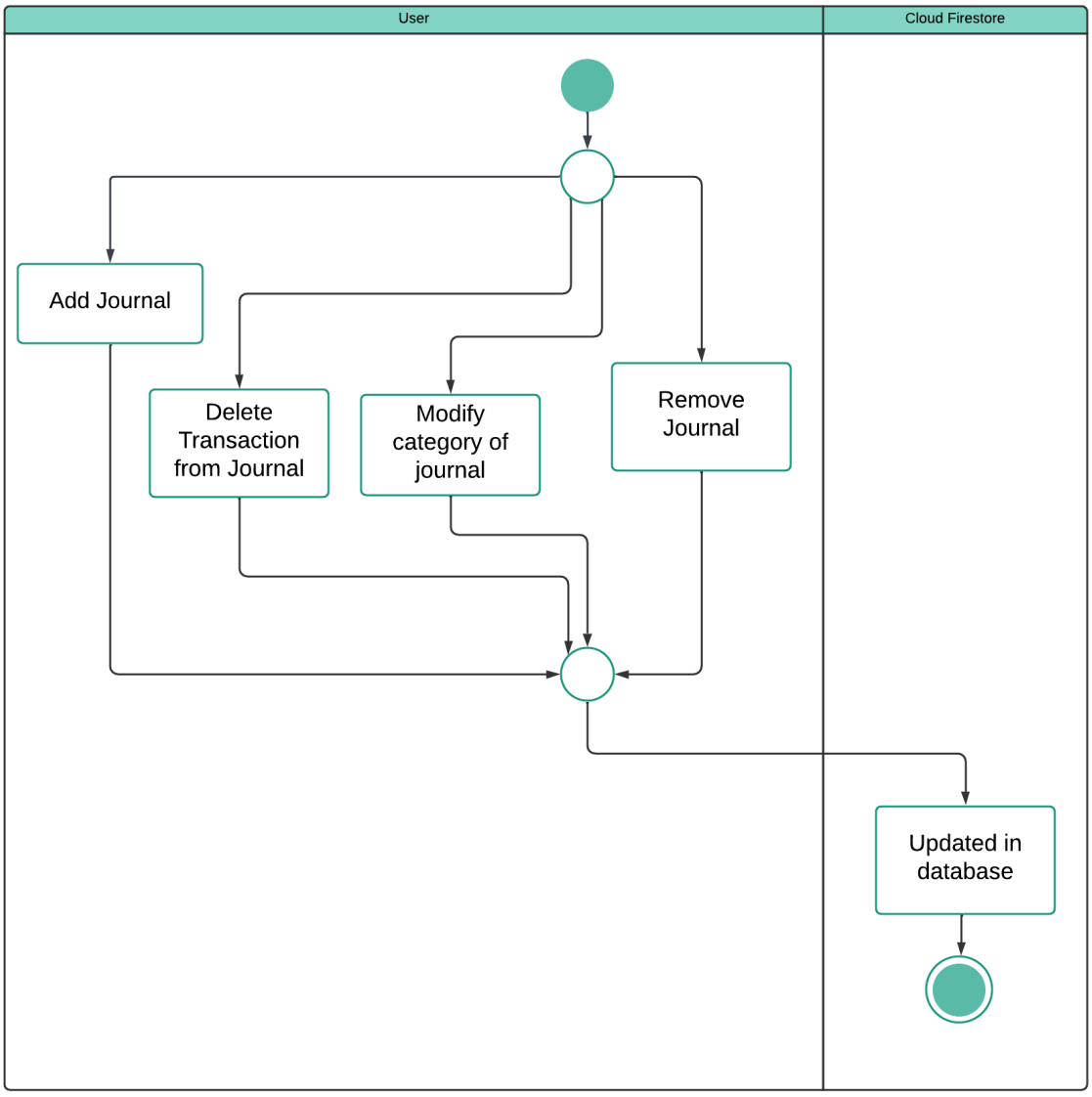
ID 4.1: Transaction Management

Reference: Use-Case Diagram 1.3



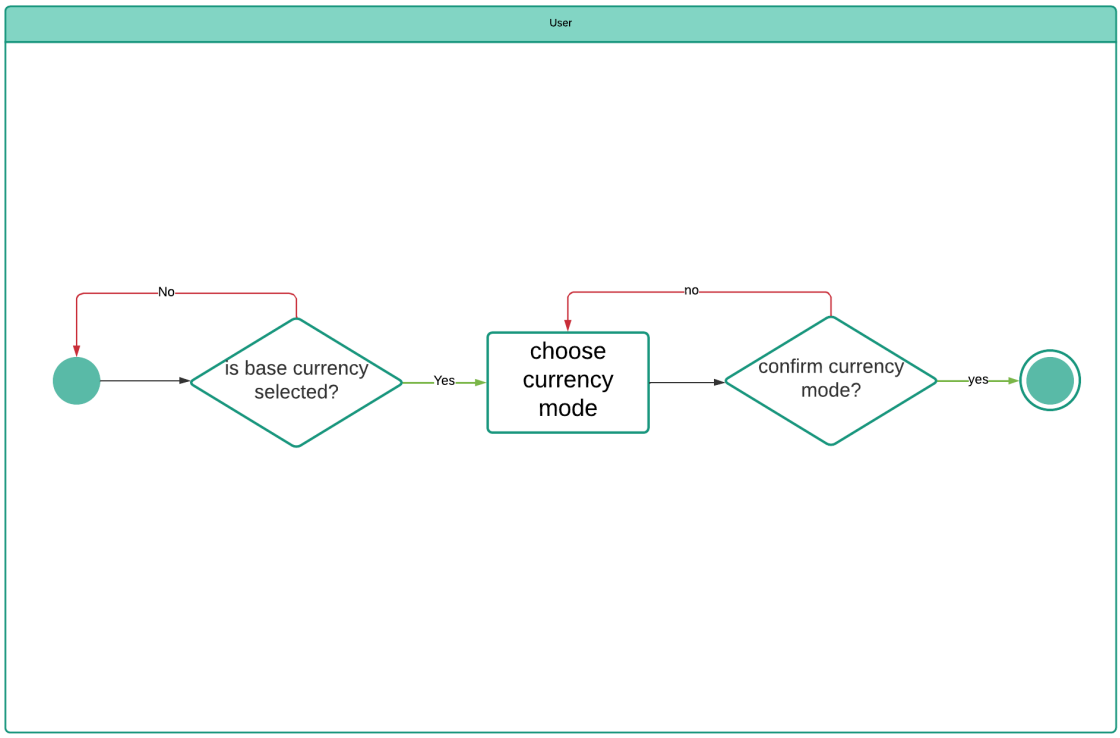
ID 4.2: Journal Management

Reference: Use-Case Diagram 1.3



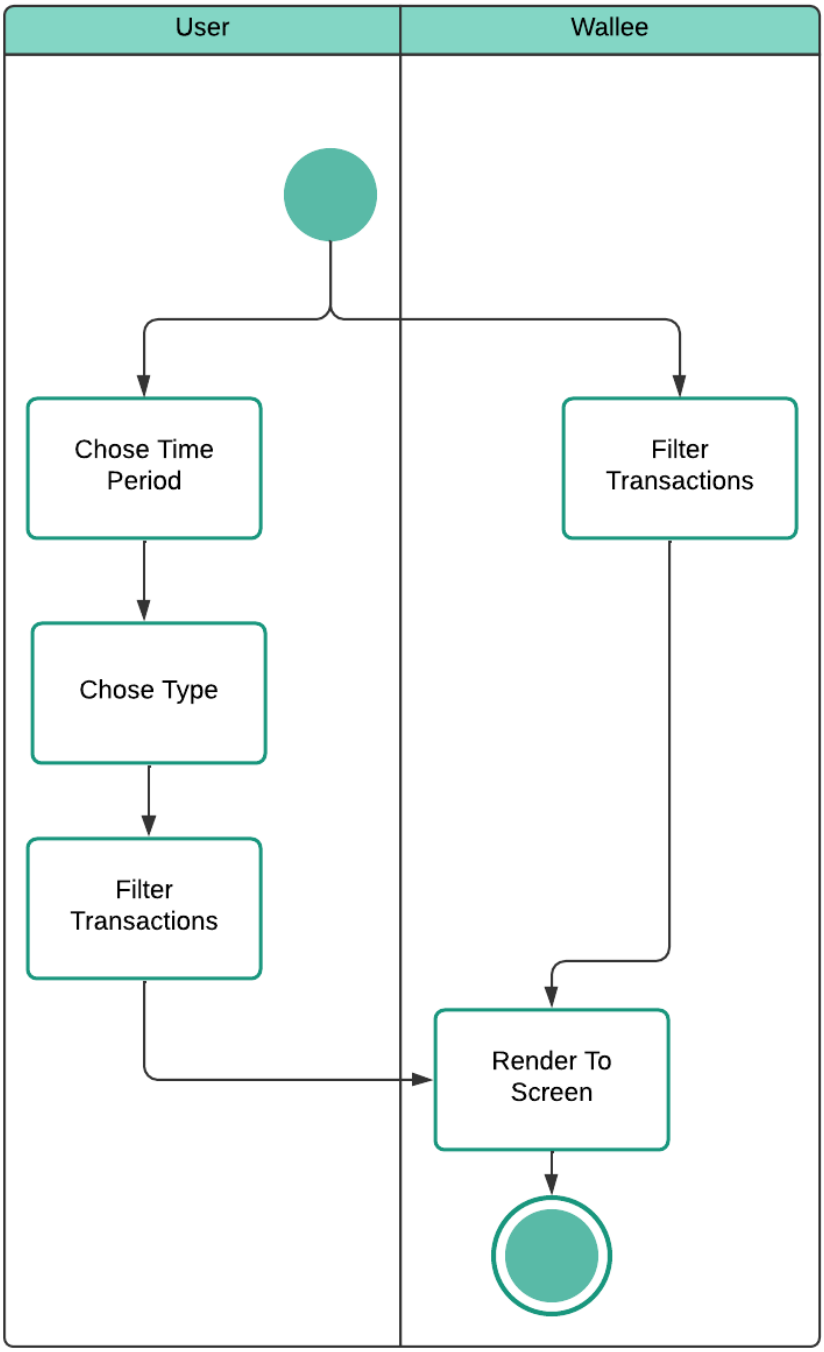
ID 5: Selection of Currency Mode

Reference: Use-Case Diagram 1.4



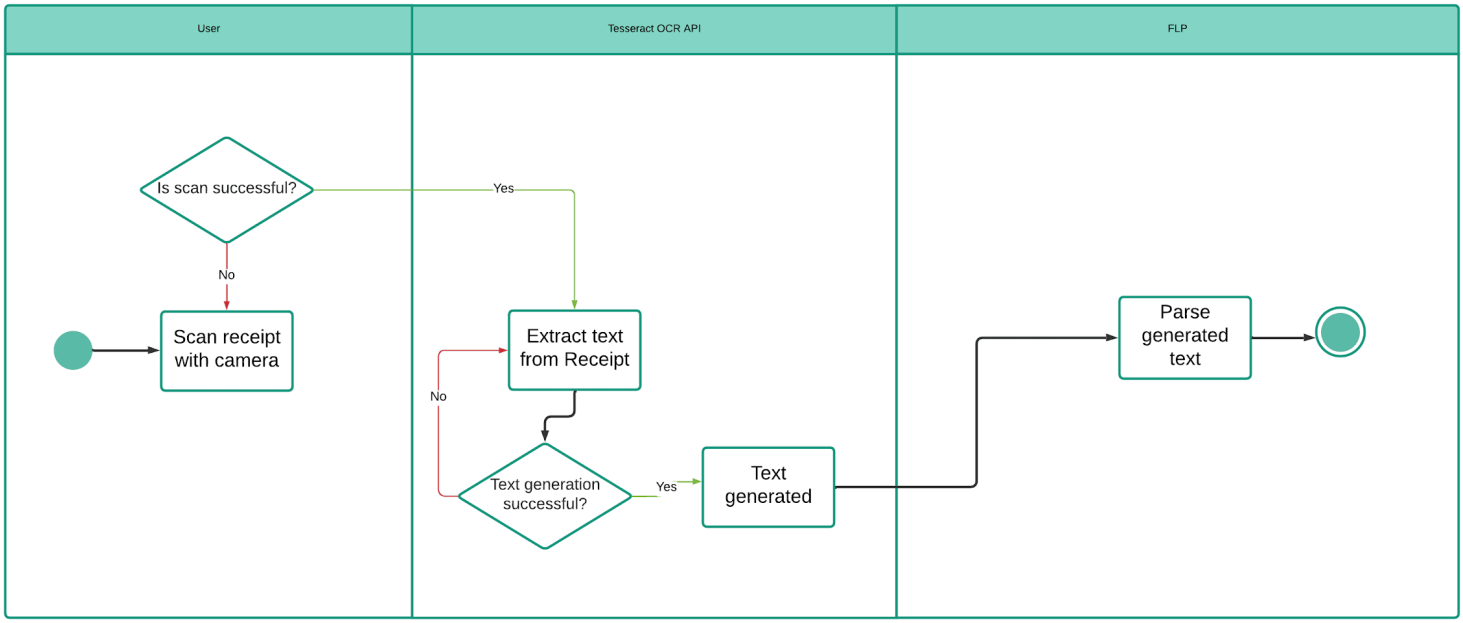
ID 6: Summary Preview

Reference: Use-Case Diagram 1.5



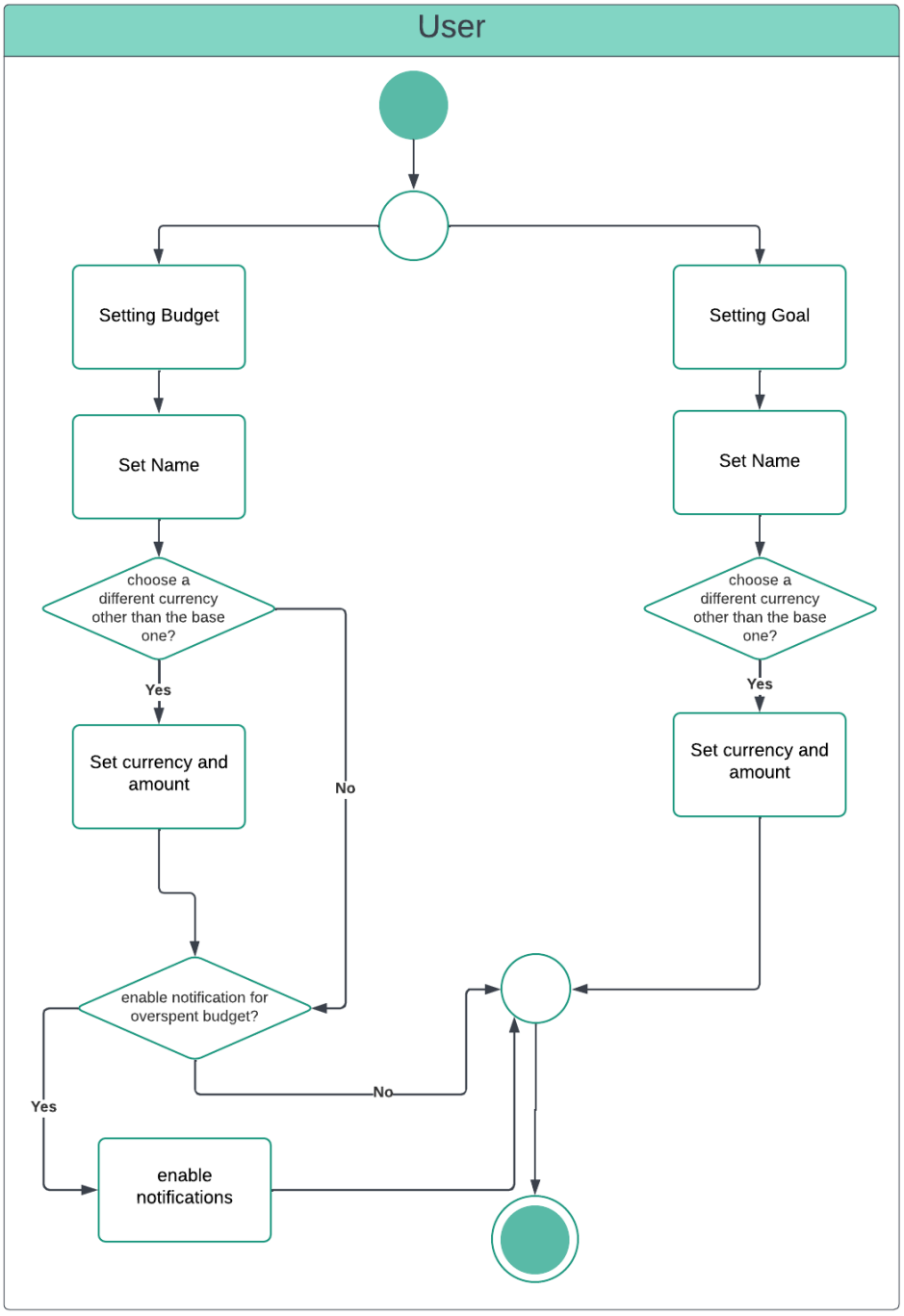
ID 7: Smart Receipt Reader

Reference: Use-Case Diagram 1.6



ID 8: Wishlist

Reference: Use-Case Diagram 1.8



5. Data Based Modeling

1. Introduction

Sometimes software requirements include the necessity to create, extend or interact with a database or complex data structures need to be constructed and manipulated. The software team chooses to create data models as a part of overall requirements modeling. The entity-relationship diagram (ERD) defines all data objects that are processed within the system, the relationships between the data objects and the information about how the data objects are entered, stored, transformed and produced within the system.

5.1. Data Objects

A data object is a representation of composite information that must be understood by the software. Here, composite information means information that has a number of different properties or attributes. A data object can be an external entity, a thing, an occurrence, a role,an organizational unit, a place or a structure.

5.2. Noun Identification

We identified all the nouns whether they are in problem space or in solution space from our usage scenario.

Table 1: Noun Identification for Data Modeling

Serial No	**Noun**	**p/s**	**Attributes**
1	Wallee	s	
2	Transaction	s	4, 50, 51, 52
3	change	p	

4	amount	s	
5	journal	s	12,10,9,14,53
6	user	s	7,19,17,20, 57
7	balance	s	
8	collection	p	
9	category	s	
10	volume	s	
11	information	p	
12	contribution	s	
13	transaction value	p	
14	Access frequency	s	
15	features	p	
16	account	p	
17	Phone number	s	
18	OTP	p	
19	name	s	
20	email	s	
21	SMS	s	
22	identifiers	s	
23	user interface	s	24, 31
24	color scheme	s	
25	financial status	p	
26	screen	p	
27	Social media	p	

28	home	p	
29	Actions menu	s	
30	history	p	
32	module	p	
33	text	p	
34	Shopping receipts	s	
35	Financial Language Processor	s	
36	Autopilot Module	p	
37	phone	p	
38	Event listener	p	
39	Local cache	p	
40	Remote database	p	
41	summary	s	
42	period	s	
43	Receipt Reader	s	
44	fields	s	
45	receipt	s	
46	expenditure	p	
47	money	p	
48	privilege	p	
49	threshold	p	
50	Associated journal	s	
51	Time of entry	s	

52	originator	s	
53	Time of last access	s	
54	Preview	s	42
55	Budget	s	60, 57, 58, 62
56	Goals	s	61, 59, 62, 57
57	currency	s	
58	Time period	s	
59	Desired date	s	
60	Budget name	s	
61	Goal name	s	
62	amount	s	

Final Data Objects

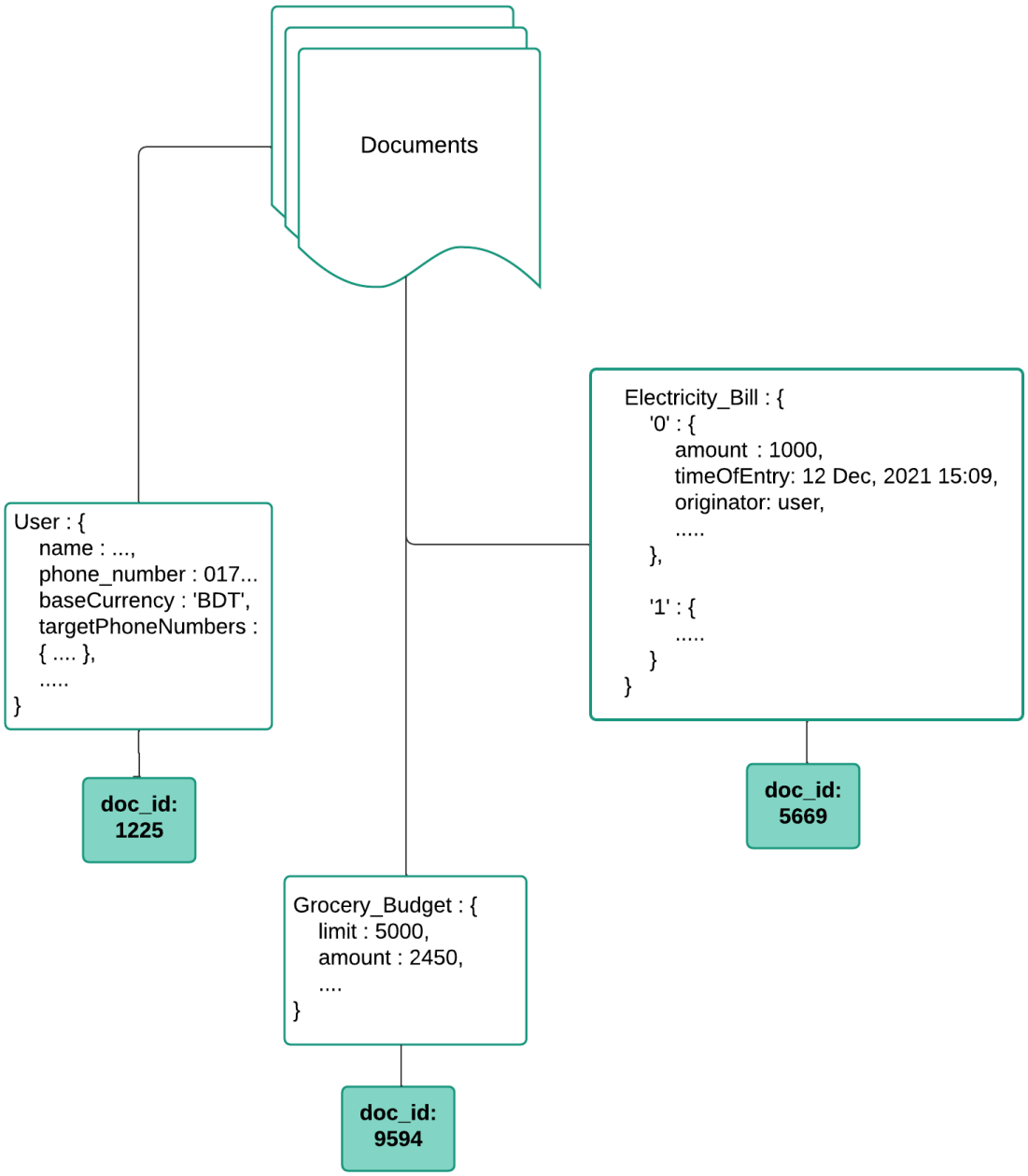
1. User
2. Transaction
3. Journal
4. Budget
5. Goal

5.3. Cloud Firestore

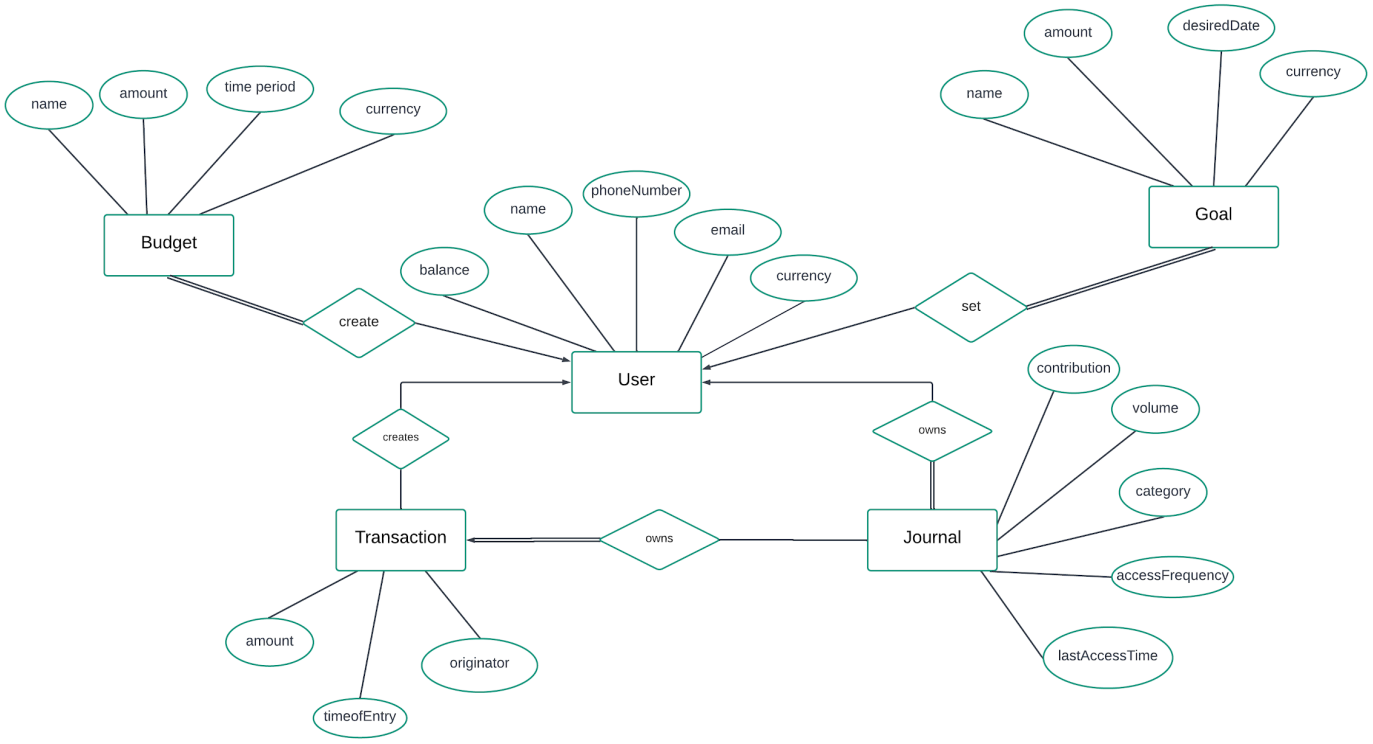
In a NoSQL Document database such as Firestore, our data objects will be stored in Documents where every field (recursively) is identified by a key-value pair. It has a large sorted index for efficient queries of a wide range of values.

A Mock Index

Value	Doc_ID
‘1000’	5669
‘user’	5669
‘017xx’	1225
‘2450’	9594
…	…
…	…



5.4. Probable ER Diagram



5.5. Schema Diagram

User		
Attributes	**Type**	**Size**
<u>userId</u>	varchar	30
name	varchar	30
balance	number	1000

phoneNumber	varchar	20
email	varchar	100
currency	varchar	100

Budget		
Attributes	**Type**	**Size**
budgetID	varchar	30
name	varchar	30
amount	number	1000
timePeriod	date	10
currency	varchar	20
userID	varchar	30

Goal		
Attributes	**Type**	**Size**
goalID	varchar	30
name	varchar	30
amount	number	1000
desiredDate	date	10
currency	varchar	20
userID	varchar	30

Transaction		
Attributes	**Type**	**Size**
transactionID	varchar	30
amount	number	1000
timeOfEntry	date	10
originator	varchar	20
userID	varchar	30

Journal		
Attributes	**Type**	**Size**
journalID	varchar	30
contribution	number	1000
volume	number	1000
category	varchar	20
accessFrequency	number	20
lastAccessTime	date	10
userID	varchar	30
transactionID	varchar	30

5.6. Database Analysis

After analyzing both database paradigms we have chosen to use Cloud Firestore for the following reasons:

- **Read efficiency over write efficiency:** Usually a transaction will be entered once, and read many times. Highly efficient read queries of FireStore should result in fast Data retrieval.

- **Works well with JavaScript:** Document based storage structure of Cloud FireStore is analogous to the JavaScript Object Notation (JSON). This will result in less overhead while developing our client side application.

- The system doesn't really require long & complicated query operations over a lot of tables.

6. Class Based Modeling

6.1. Class based modeling concept

Class-based modeling is a stage of requirements modeling. It uses common concepts of object-oriented programming to craft an impression of an application that can be understood by nontechnical stakeholders. Class-based modeling represents: -

- The objects the system will manipulate
- The operation that will be applied for effective manipulation
- The relationships between the objects
- The collaborations that occur between the classes.

6.2. Identifying Classes

Classes are identified by underlining each noun or noun phrase and plotting it into a simple table. If the class (noun) is required to implement a solution, then it becomes a part of the solution space. Otherwise if the noun is used only to describe a solution, it is regarded as a part of the problem space. Once all the nouns have been isolated, General classification and Selection criteria is done.

6.3. General Classification

Nouns belonging to the solution space should exhibit any of the following criteria to be considered as a class. The 7 general characteristics are stated below:

1. **External entities:** (e.g., other systems, devices, people) that produce or consume information to be used by a computer-based system.

2. **Things:** (e.g., reports, displays, letters, signals) that are part of the information domain for the problem.

3. **Occurrences or events:** (e.g., a property transfer or the completion of a series of robot movements) that occur within the context of system operation.

4. **Roles:** (e.g., manager, engineer, salesperson) played by people who interact with the system.

5. **Organizational units:** (e.g., division, group, team) that are relevant to an application.

6. **Places:** (e.g., manufacturing floor or loading dock) that establish the context of the problem and the overall function of the system.

7. **Structures** (e.g., sensors, four-wheeled vehicles, or computers) that define a class of objects or related classes of objects.

Noun	**General Classification**
Wallee	7
Transaction	2, 3, 7
amount	2,
journal	2, 3, 7
user	4, 5, 7
balance	2
category	2
volume	2
contribution	2
Access frequency	2
Phone number	2

name	2
email	2
SMS	1, 3
identifiers	2, 3
color scheme	2, 3
Actions menu	2, 3
Shopping receipts	1, 2
Financial Language Processor	2, 3, 7
summary	2
period	2
Receipt Reader	7
fields	2
receipt	1, 2
Budget	2, 3, 7
Associated journal	2, 3
Time of entry	2
originator	4
Time of last access	2
Currency	2, 5
Goal	2, 3, 7

6.4. Selection Criteria

The six selection characteristics that should be used as you consider each potential class for inclusion in the analysis model:

1. Retained information: The potential class will be useful during analysis only if information about it must be remembered so that the system can function.

2. Needed services: The potential class must have a set of identifiable operations that can change the value of its attributes in some way.

3. Multiple attributes: During requirement analysis, the focus should be on "major" information; a class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.

4. Common attributes: A set of attributes can be defined for the potential class and these attributes apply to all instances of the class.

5. Common operations: A set of operations can be defined for the potential class and these operations apply to all instances of the class.

6. Essential requirements: External entities that appear in the problem space and produce or consume information essential to the operation of any solution for the system will almost always be defined as classes in the requirements modeling.

Noun	**Selection Criteria**
Transaction	1, 2, 3, 4, 5
journal	1, 2, 3, 4, 5
User	1, 2, 3, 4, 5
SMS	6

identifiers	1
user interface	1
color scheme	1
Actions menu	1
Shopping receipts	6
Financial Language Processor	1, 2, 3, 4, 5
receipt	1, 2
Associated journal	1
Currency	1, 2
Preview	1, 2, 3 ,4, 5
Budget	1, 2, 3, 4, 5
Goal	1, 2, 3, 4, 5

6.5. List of verbs

Verb	**Description**
increase	Increase user's balance
decrease	Decrease user's balance
Create	Create Transaction
	Create Account
	Create Budget
add	Add email address
	Add profile photo

	Add base currency
	Add transactions entities
delete	Delete transaction entities
Change	Change contribution
Set	Set category
have	Exchange rates
process	Process transaction
send	Send OTP
provide	Provide OTP
	Provide name
	Provide balance
manipulate	Manipulate journals
change	Change phone number
	Change currency
specify	Specify summary period
choose	Choose summary preview mode
map	Map to a particular color
have	Have specific colors
track	Track appropriate journal entry
search	Search through journals
sort	Sort Journals
modify	Modify Journal
parse	Parse SMS
scan	Scan receipts

record	Record transactions
set	Set Goals

Selected Classes:

1. User
2. Transaction
3. Income : Journal
4. Expense : Journal
5. Budget
6. Goal
7. Chart: Preview
8. Piechart: Preview
9. Bargraph : Preview
10. Currency
11. ReceiptView
12. ColorScheme
13. Balance

6.6. Attributes and Methods Identification

Name	**Attributes**	**Methods**

User	-name -email -phoneNumber -Currency currency -balance -baseCurrency -journalList -profilePhoto -transactionList -budgetList -goalList	+sendAuthRequest() +pinJournal() +setEmail() +setProfilePhoto() +setCurrency() +setBalance() +createJournal() +modifyJournalCategory() +createTransaction() +deleteTransaction() +changePhoneNumber() +createBudget() +setGoal() +searchJournal() +sortJournal()
CurrencyMode	-currentCurrency -cachedCurrencyList	-updateCache() +getExchangeRate() +convertToCurrency()
Transaction	-amount -associatedJournal -currency -timeOfEntry -lastAccessTime -originator	+getAmount() +getCurrency() +editAmount() +insertToJournal() +revoke()
<<Journal>>	-transactionVolume -accessFrequency -lastAccessTime -category	+sortTransactions() +filterTransactions() +removeTransaction() +getContribution() +getAccessTime() +getAccessFrequency() +setCategory() +contributeToBalance()

IncomeJournal : Journal	- transactionsList -contribution -transactionVolume -accessFrequency -lastAccessTime -category -associatedBudget	+sortTransactions() +filterTransactions() +removeTransaction() +getContribution() +getAccessTime() +getAccessFrequency() +setCategory() +contributeToBalance()
ExpenseJournal : Journal	-transactionsList -contribution -transactionVolume -accessFrequency -lastAccessTime -category -associatedBudget	+sortTransactions() +filterTransactions() +removeTransaction() +getContribution() +getAccessTime() +getAccessFrequency() +setCategory() +contributeToBalance()
Budget	-category -associatedJournal -amount -timePeriod	+setBudget() +getBudget()
Goal	-category -associatedJournal -amount -desiredDate	+setGoal() +getGoal()
<<Preview>>	-timeRange -transActionsList	+renderToScreen() +filter()

Chart : Preview	-timeRange -transActionsList	+renderToScreen() +filter()
PieChart : Preview	-timeRange -transactionsList -segmentCount	-generateSegment +renderToScreen() +filter()
BarGraph : Preview	-timeRange -transactionsList -barCount	-generateBar() +render() +filter()
ReceiptView	-proposedTransactions	+createTransaction() +assignToJournal() +selectJournalForAll() +revokeTransaction()
ColorScheme	-incomeColors -expenseColors -colorMap -userBalanceGradient	+generateUIColor() +applyToUI() +mapColorToJournal() +pollUserBalance()

6.7. Class Cards

User	
Attributes	**Methods**
-name -email -phoneNumber -defaultCurrency -balance -baseCurrency -journalList -profilePhoto -transactionList -budgetList -goalList	+sendAuthRequest() +pinJournal() +setEmail() +setProfilePhoto() +setCurrency() +setBalance() +createJournal() +modifyJournalCategory() +createTransaction() +deleteTransaction() +changePhoneNumber() +createBudget() +setGoal() +searchJournal() +sortJournal()
Responsibilities	**Collaborators**
• Add email address and profile picture • Add base currency • Manages(Add or delete) transaction entities • Choose summary Preview Mode • Search and sort Journals • Sort Journals • Modify journal Category • Create Budget • Set goals • Keep track of user's balance	Transaction, Budget, Preview, CurrencyMode, Goal, Journal

CurrencyMode	
Attributes	**Methods**
-currencyName -country	-updateCache() +getExchangeRate()

-currencyCode	+convertToCurrency()
Responsibilities	**Collaborators**
• Have exchange rates	User, Transaction, Budget, Goal,

Transaction	
Attributes	**Methods**
-amount -associatedJournal -timeOfEntry -lastAccessTime -originator	+getAmount() +getCurrency() +editAmount() +changeOriginator() +assignToJournal() +revoke()
Responsibilities	**Collaborators**
• Serve as the atomic unit for recording the user's financial status.	Journal, CurrencyMode, ReceiptReview

Journal	
Attributes	**Methods**
-type -transactionsList -contribution -transactionVolume -accessFrequency -lastAccessTime -category	+sortTransactions() +filterTransactions() +removeTransaction() +getContribution() +getAccessTime() +setCategory() +contributeToBalance()
Responsibilities	**Collaborators**
• Increase or decrease user’s	Transaction, CurrencyMode, Budget

balance	

Budget	
Attributes	**Methods**
-category -amount -currency -timePeriod	+setBudget() +getBudget()
Responsibilities	**Collaborators**
• Keep track of budget	Journal, CurrencyMode

PreviewMode	
Attributes	**Methods**
-timeRange -transactionsList	+render() +filter()
Responsibilities	**Collaborators**
• Show a structured overview of transactions over a period of time	User, Transaction, Journal

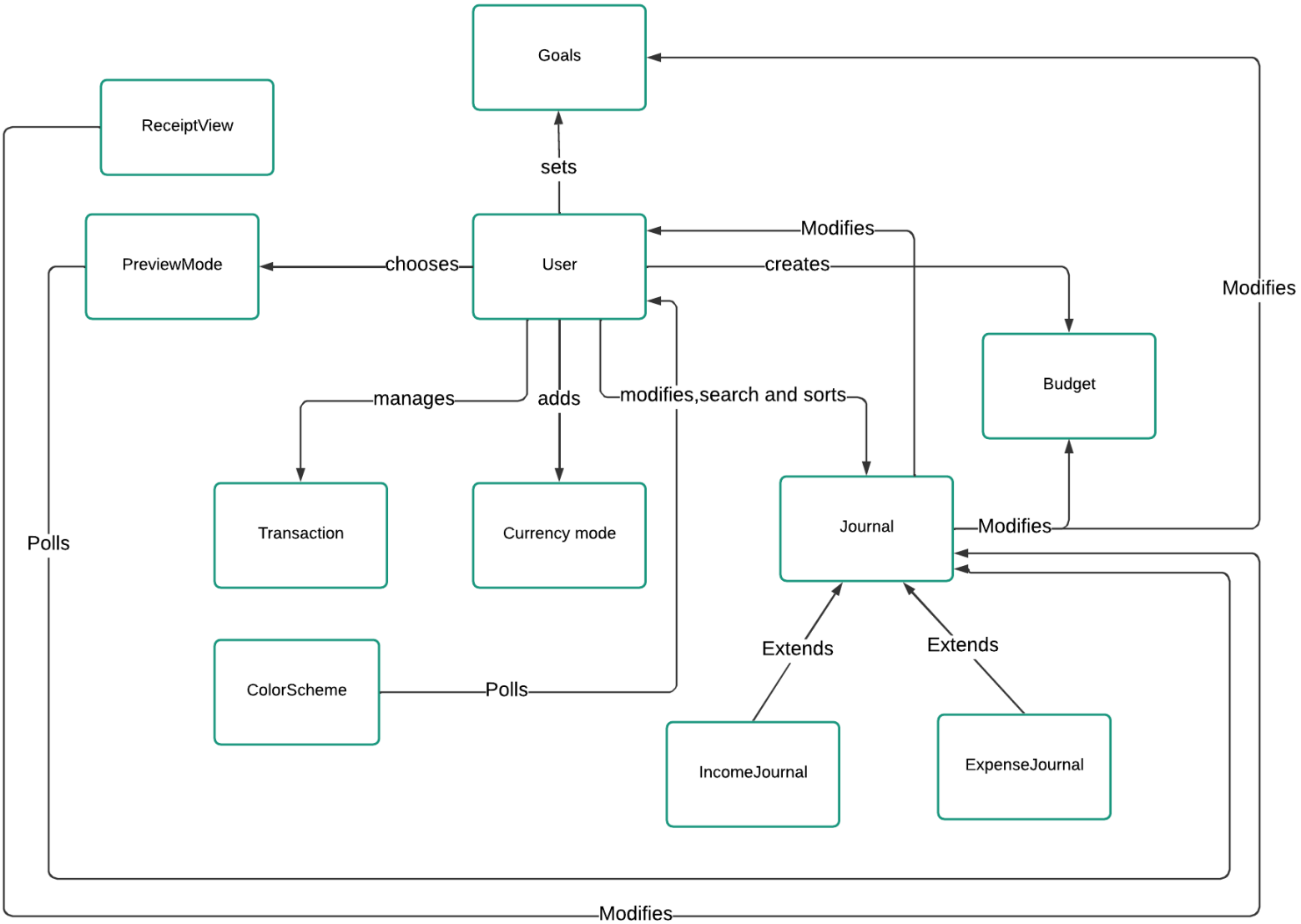
ReceiptView	
Attributes	**Methods**
-proposedTransactionsList	+createTransaction() +selectJournalForAll() +revokeTransaction()

Responsibilities	Collaborators
• Show List of proposed transactions extracted from a receipt	Transaction

ColorScheme	
Attributes	**Methods**
-incomeColors -expenseColors -colorMap -userBalanceGradient	+generateColor() +applyToUI() +mapColorToJournal() +pollUserBalance()
Responsibilities	**Collaborators**
• Map color to journals and UI • Render Adaptive Theme	Journal

Goal	
Attributes	**Methods**
-name -amount -currency -desiredDate	+setGoal() +getGoal()
Responsibilities	**Collaborators**
	User

6.8. CRC Diagram



7. Behavioral Modeling

7.1. List of Events

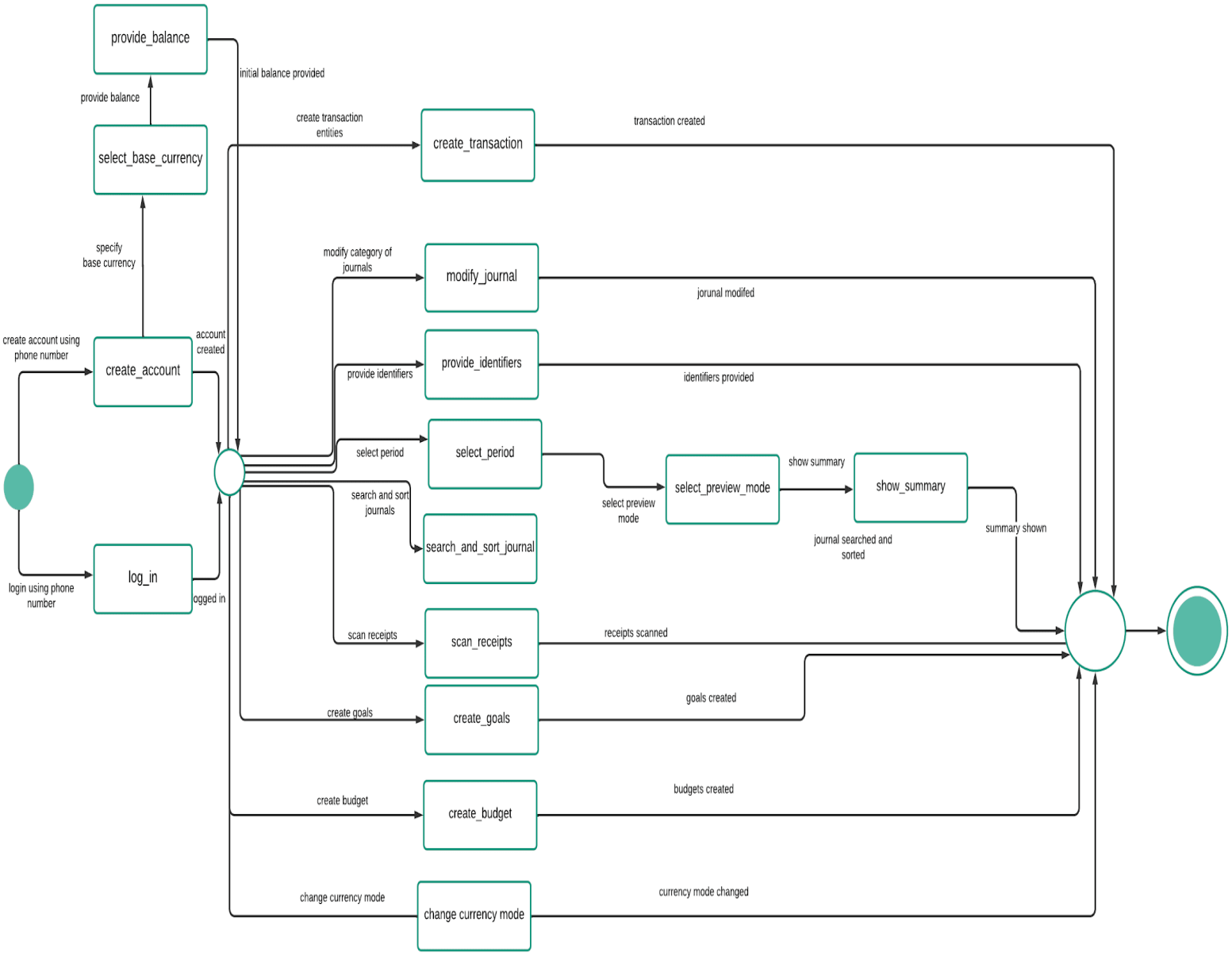
Event	Initiator	Collaborator	State Name
Modify user balance	Journal	User	modify_balance
Create Account	User	Firebase Auth Services	create_account
Log in	User	Firebase Auth Services	
Create Transaction	User, System	Transaction	create_transaction
Create Budget	User	Budget	create_budget
Add base currency	User	CurrencyMode	add_base_currency
Modify transaction entities	User	Transaction	modify_transaction_entity
Create journal	User, System	Journals	create_journal
Track appropriate journals	FLP	Journal	find_appropriate_journal

Provide current balance	User		provide_current_balance
Modify journals	User	Journal	modify_journals
Select base currency	User	currencyMode	select_base_currency
change currency mode	User	currencyMode	changeCurrencyMode
View summary	User	Journals	view_summary
Choose preview mode	User	Preview	choose_preview_mode
Map colors to journals	ColorScheme	Journal	map_colors_to_journals
Search through journals	User	Journal	search_journal
Provide identifiers	User		provide_identifiers
Sort journals	User	Journal	sort_journal
Get Exchange Rate	CurrencyMode		get_exchange_rate

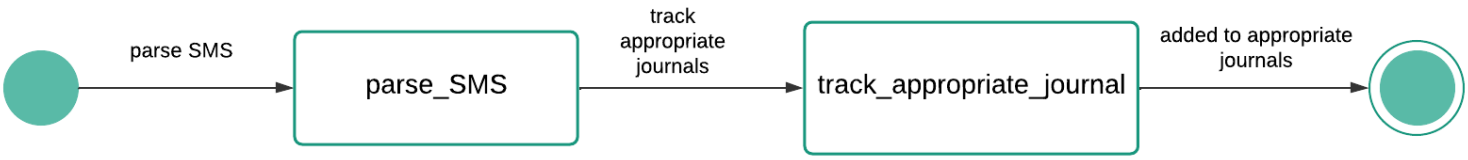
Update Currency Cache	CurrencyMode		updated_cache
parse SMS	FLP		parse_SMS
scanReceipts	User	ReceiptViewer	scan_receipts
Set Goals	User	Goal	set_goals
Update Balance	Journal	Balance	update_balance
Update ColorScheme	Balance	ColorScheme	update_color_scheme
Poll Journal	PreviewMode	Journal	poll_journal

7.2. State Transition Diagrams:

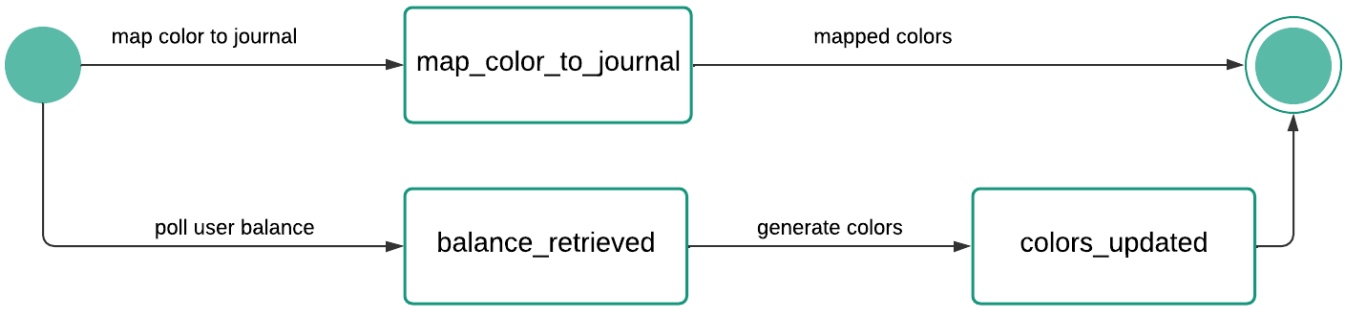
ID 1 : User



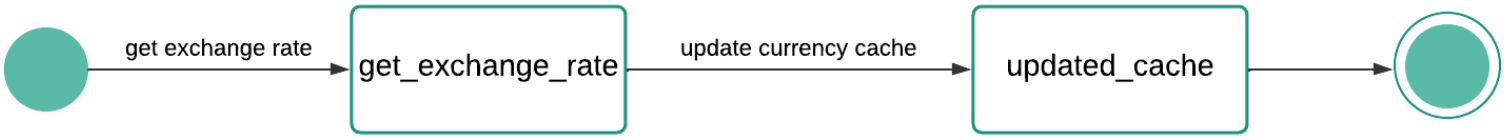
ID 2 : FLP



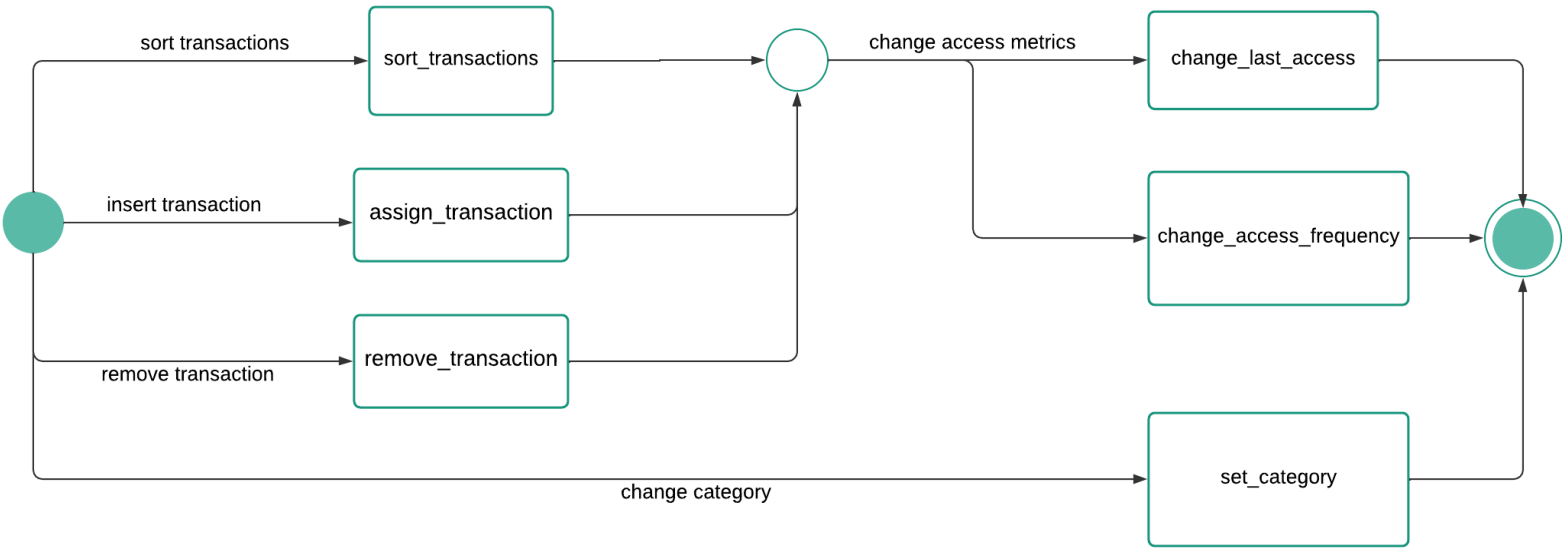
ID 3 : ColorScheme



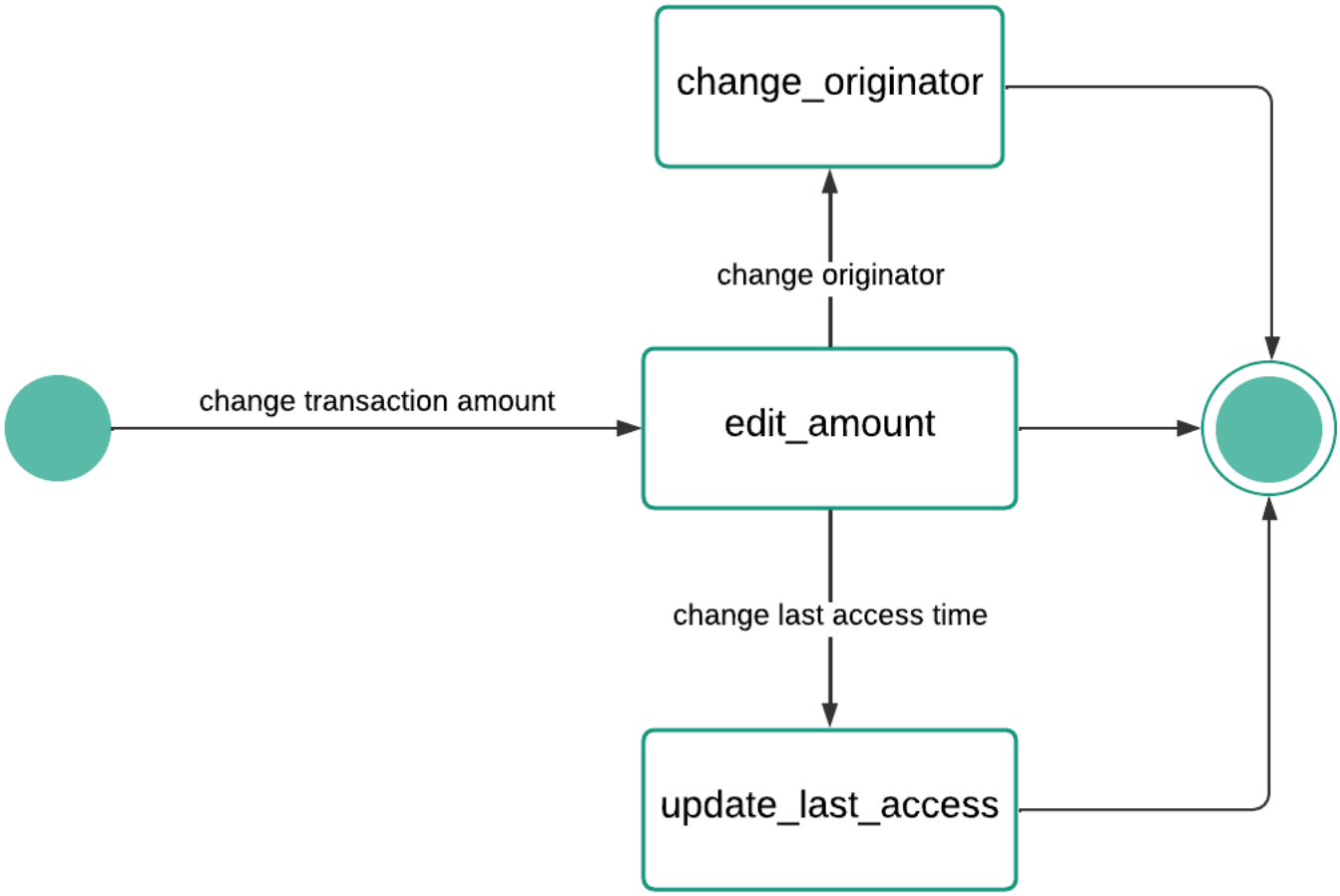
ID 4 : CurrencyMode



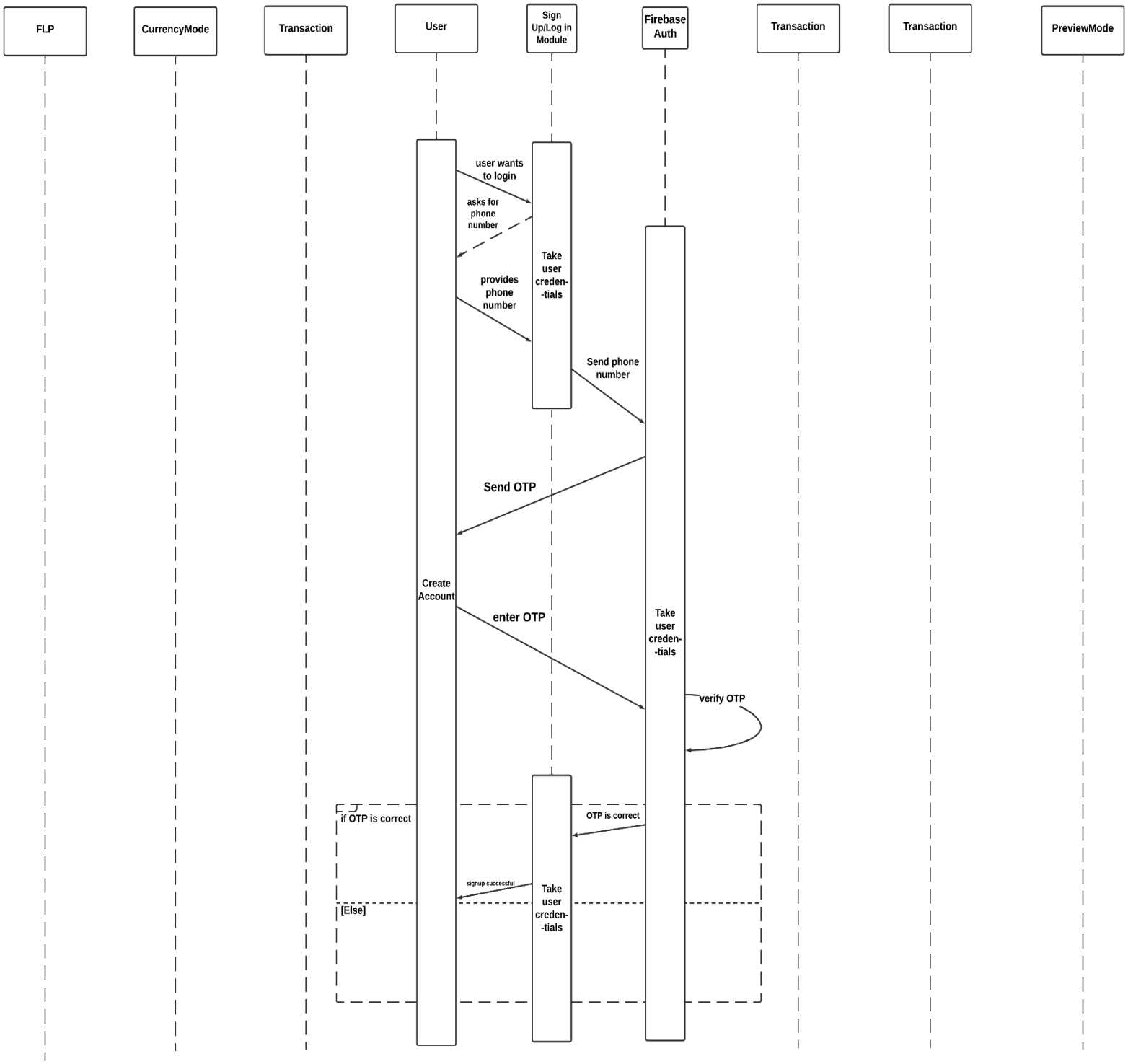
ID 5 : Journal

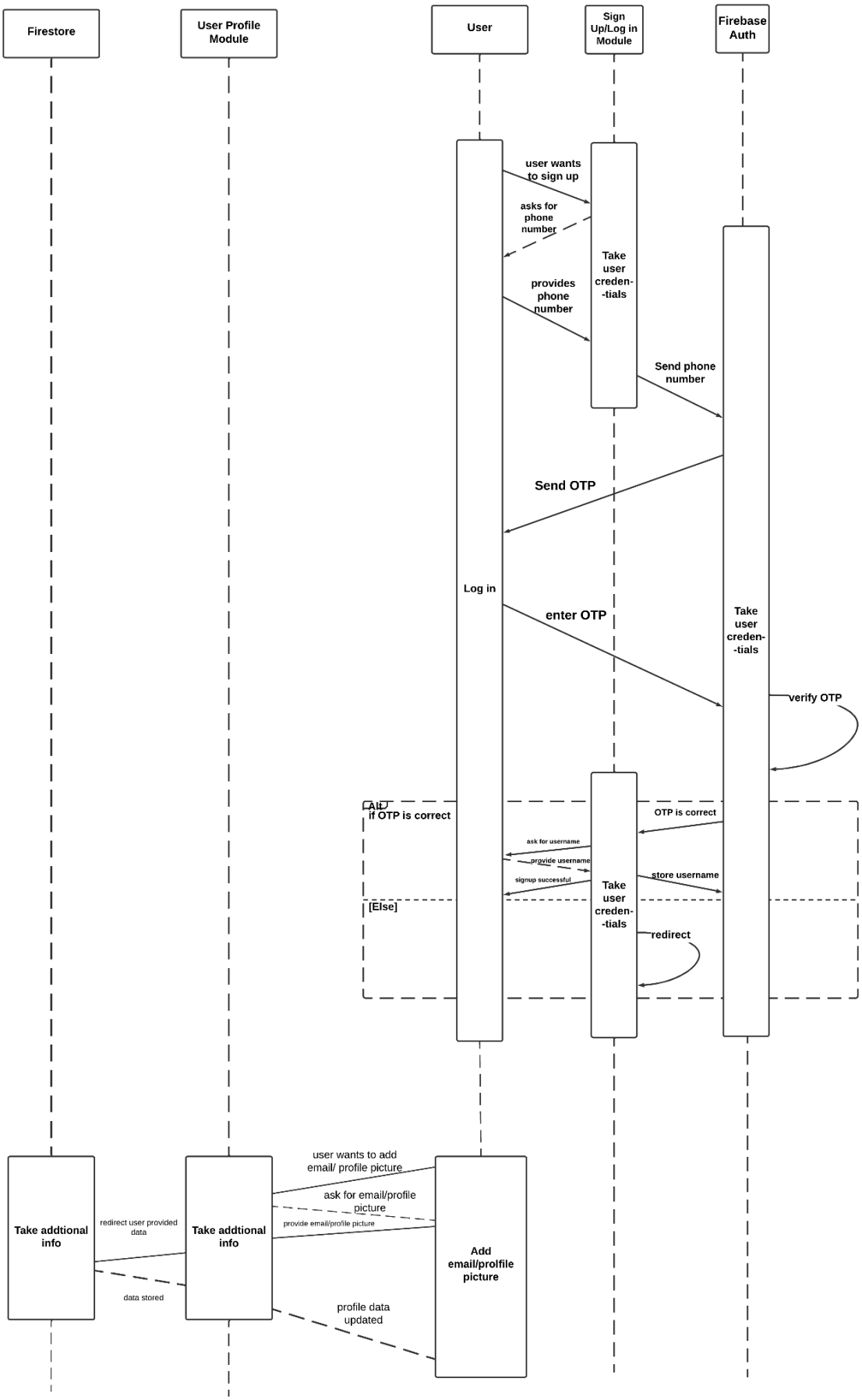


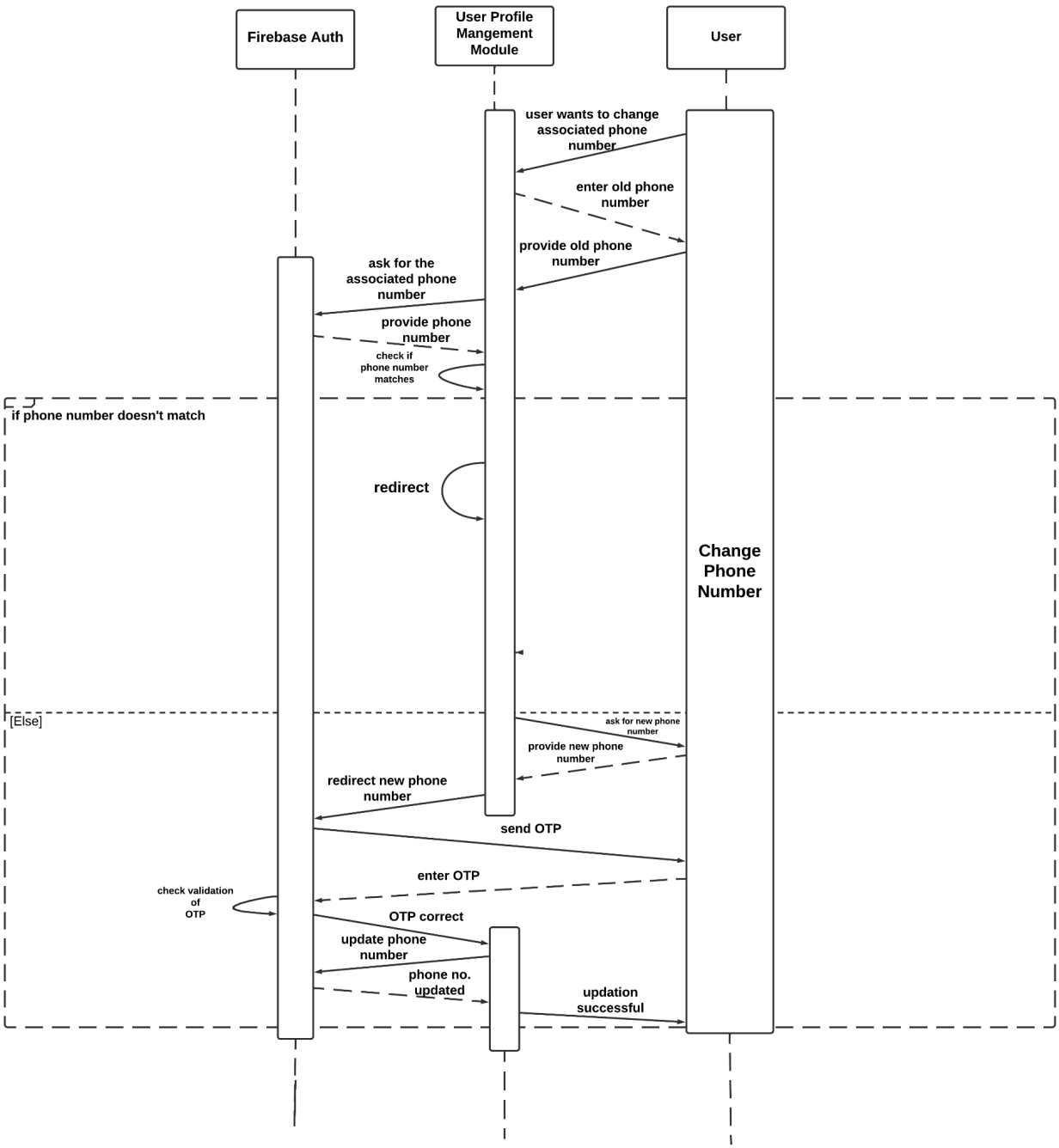
ID 6 : Transaction

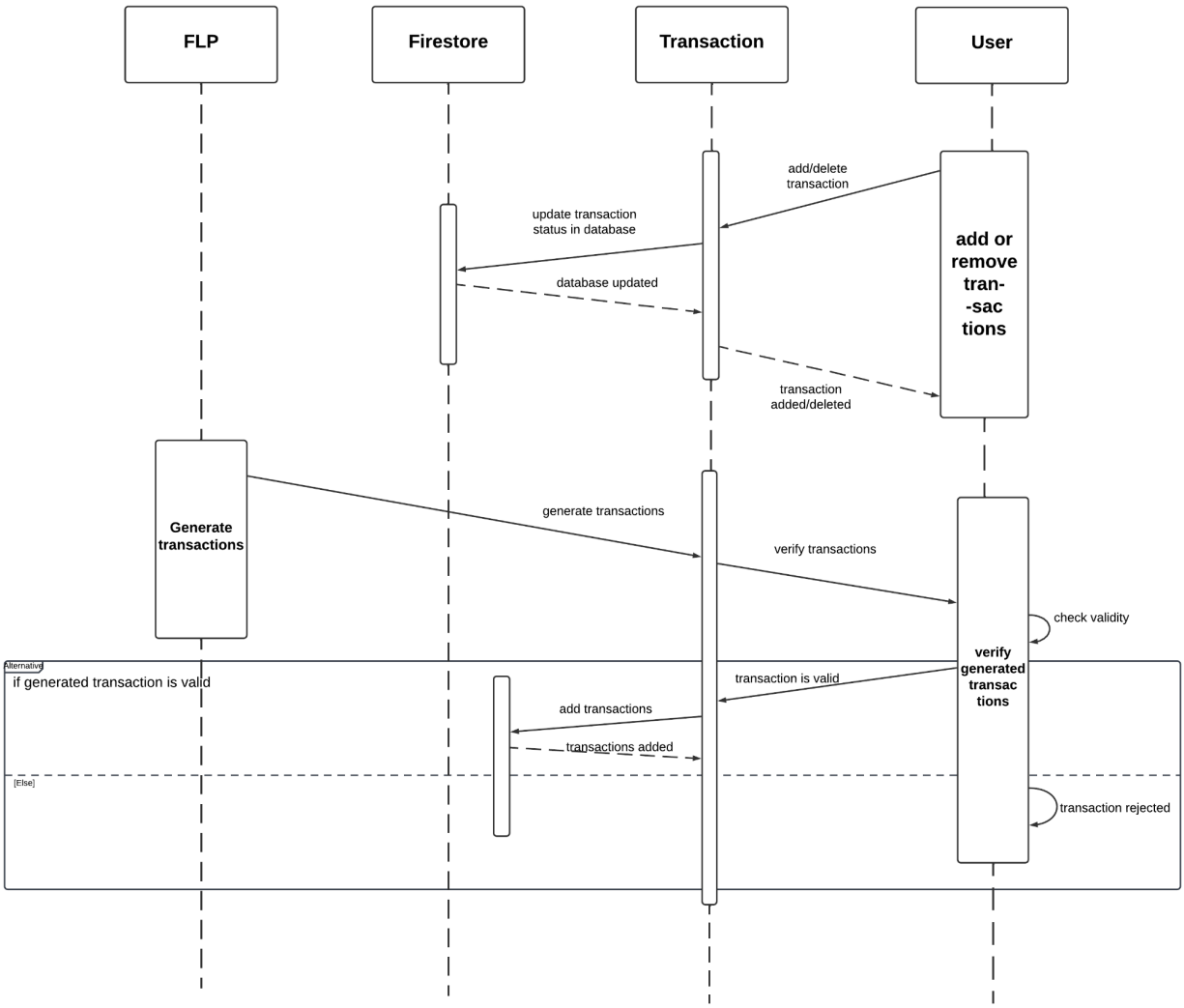


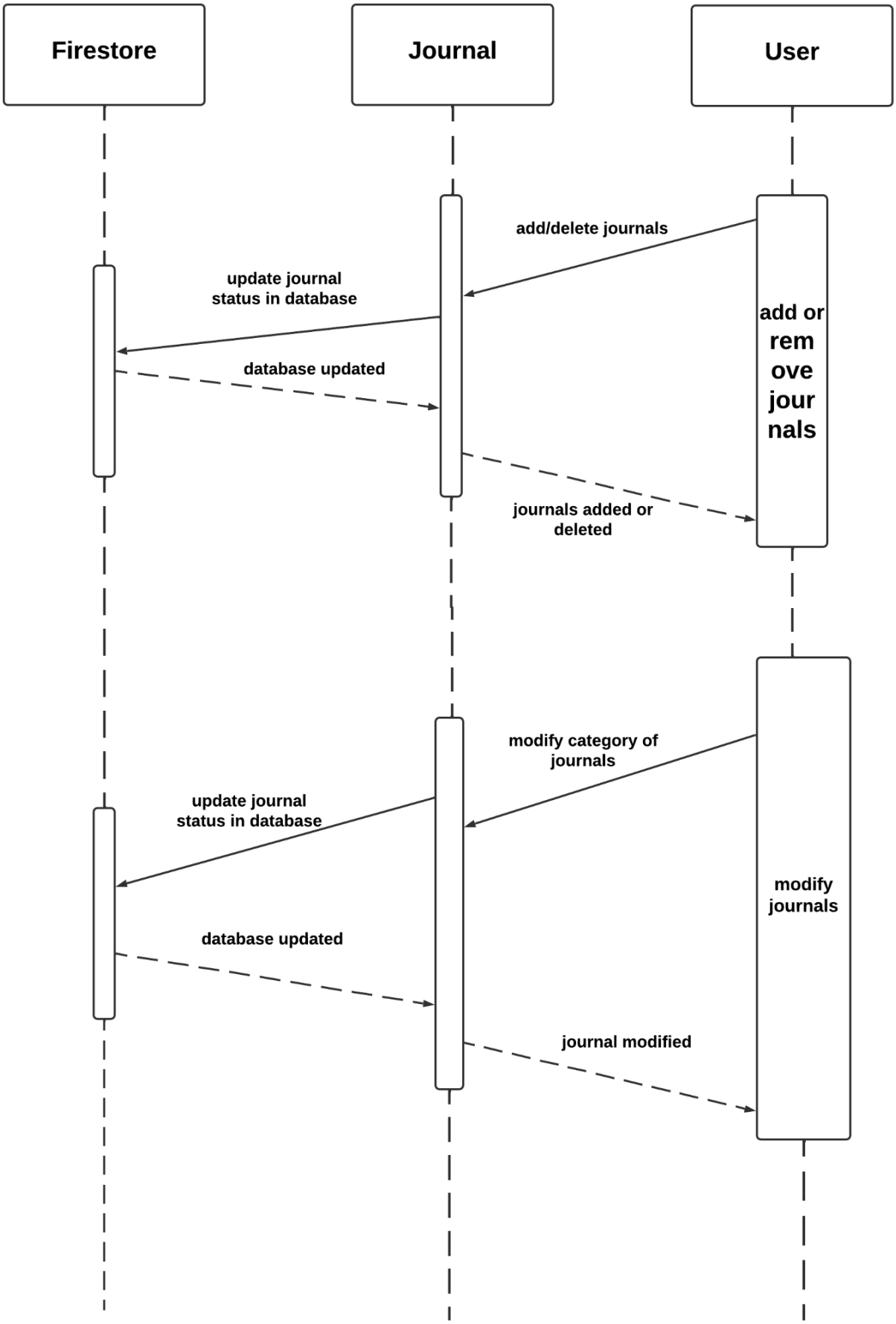
8. Sequence Diagrams:

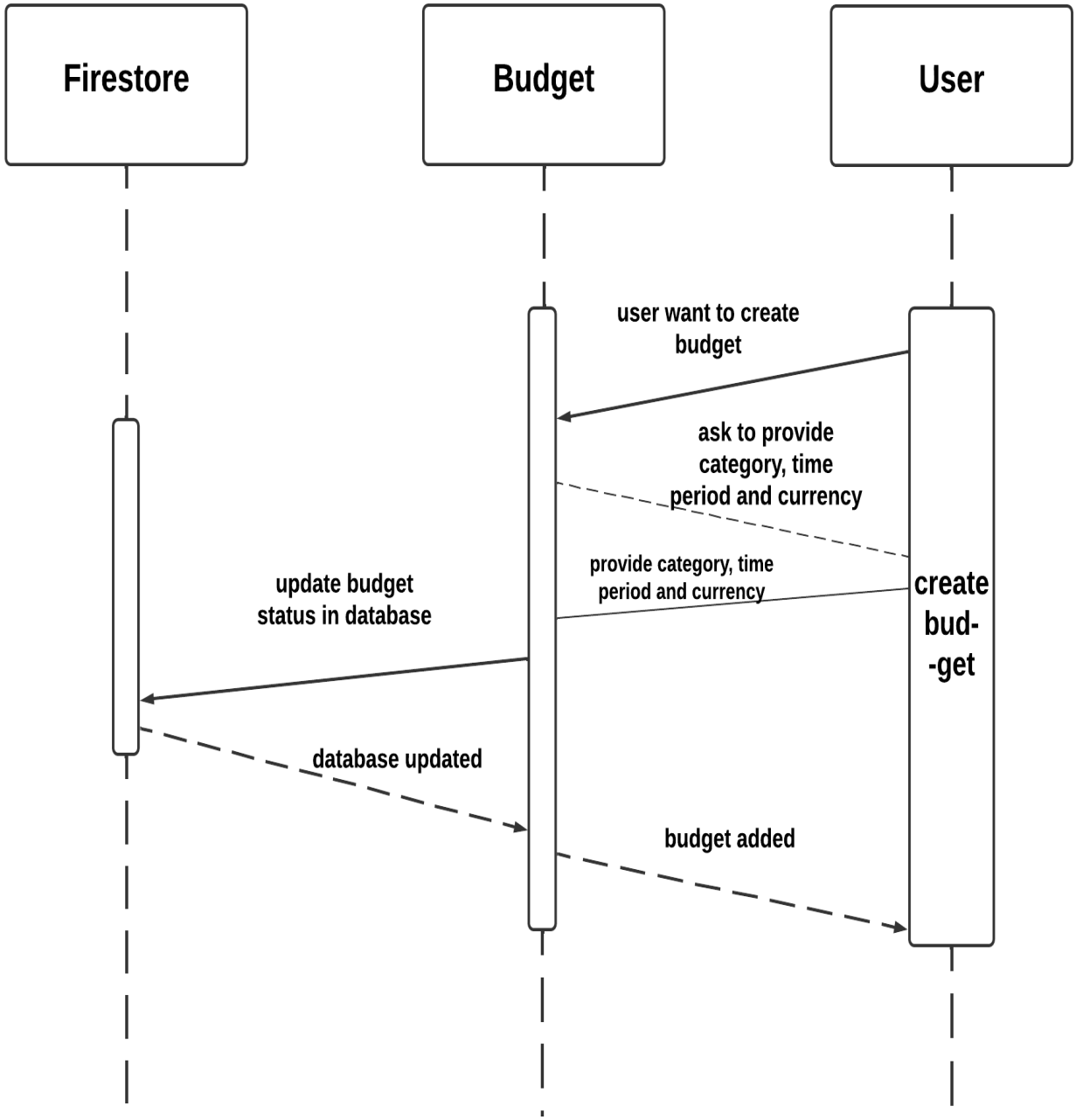


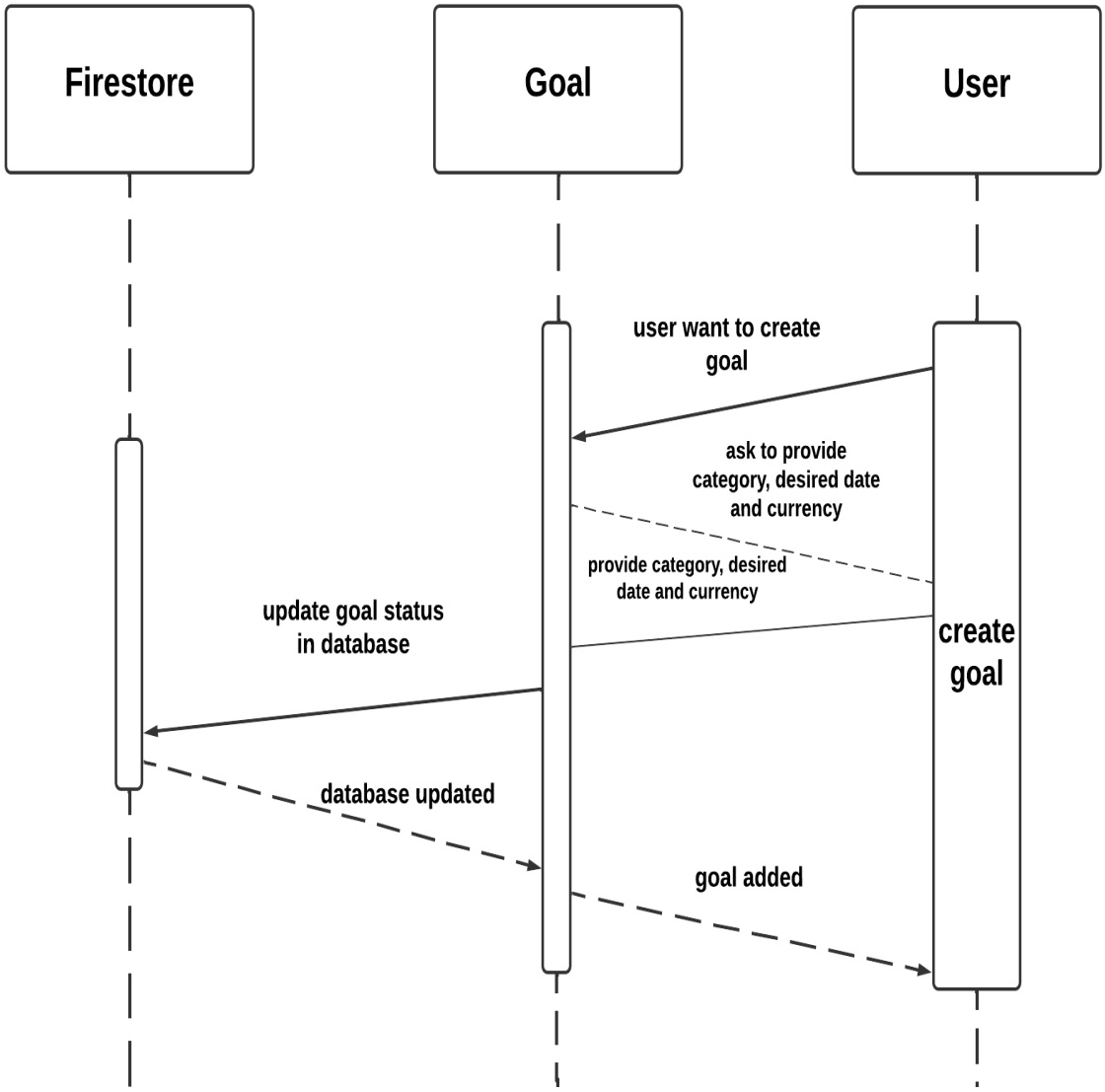

Firestore
User Profile Module
User
Sign Up/Log in Module
Firebase Auth
user wants to sign up
asks for phone number
Take user creden-tials
provides phone number
Send phone number
Send OTP
Log in
enter OTP
Take user creden-tials
verify OTP
Alt
if OTP is correct
OTP is correct
ask for username
provide username
signup successful
store username
Take user creden-tials
[Else]
redirect
Take addtional info
redirect user provided data
Take addtional info
user wants to add email/ profile picture
ask for email/profile picture
provide email/profile picture
Add email/prolfile picture
data stored
profile data updated

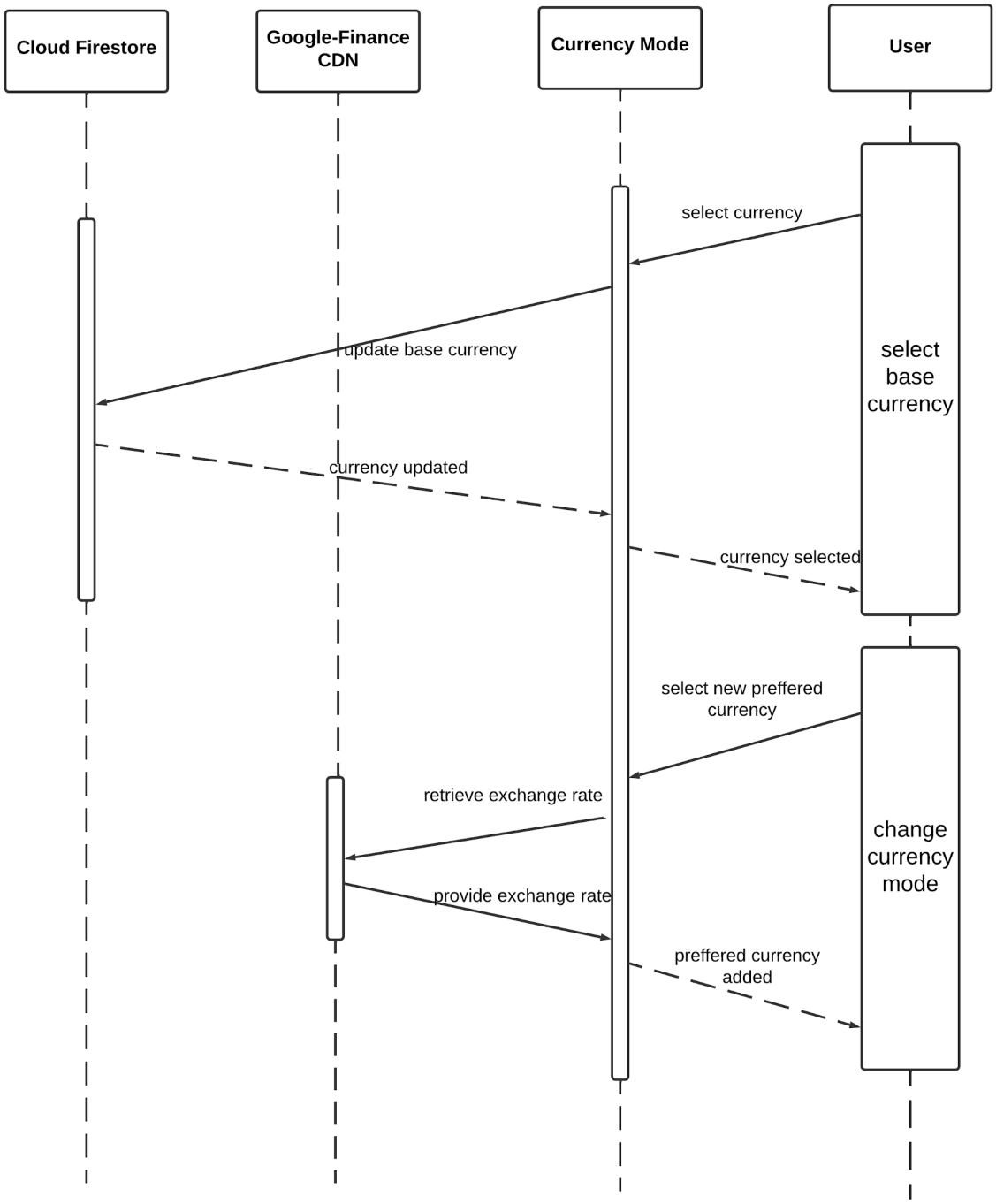

Firebase Auth
User Profile Mangement Module
User
user wants to change associated phone number
enter old phone number
provide old phone number
ask for the associated phone number
provide phone number
check if phone number matches
if phone number doesn't match
redirect
Change Phone Number
[Else]
ask for new phone number
provide new phone number
redirect new phone number
send OTP
enter OTP
check validation of OTP
OTP correct
update phone number
phone no. updated
updation successful

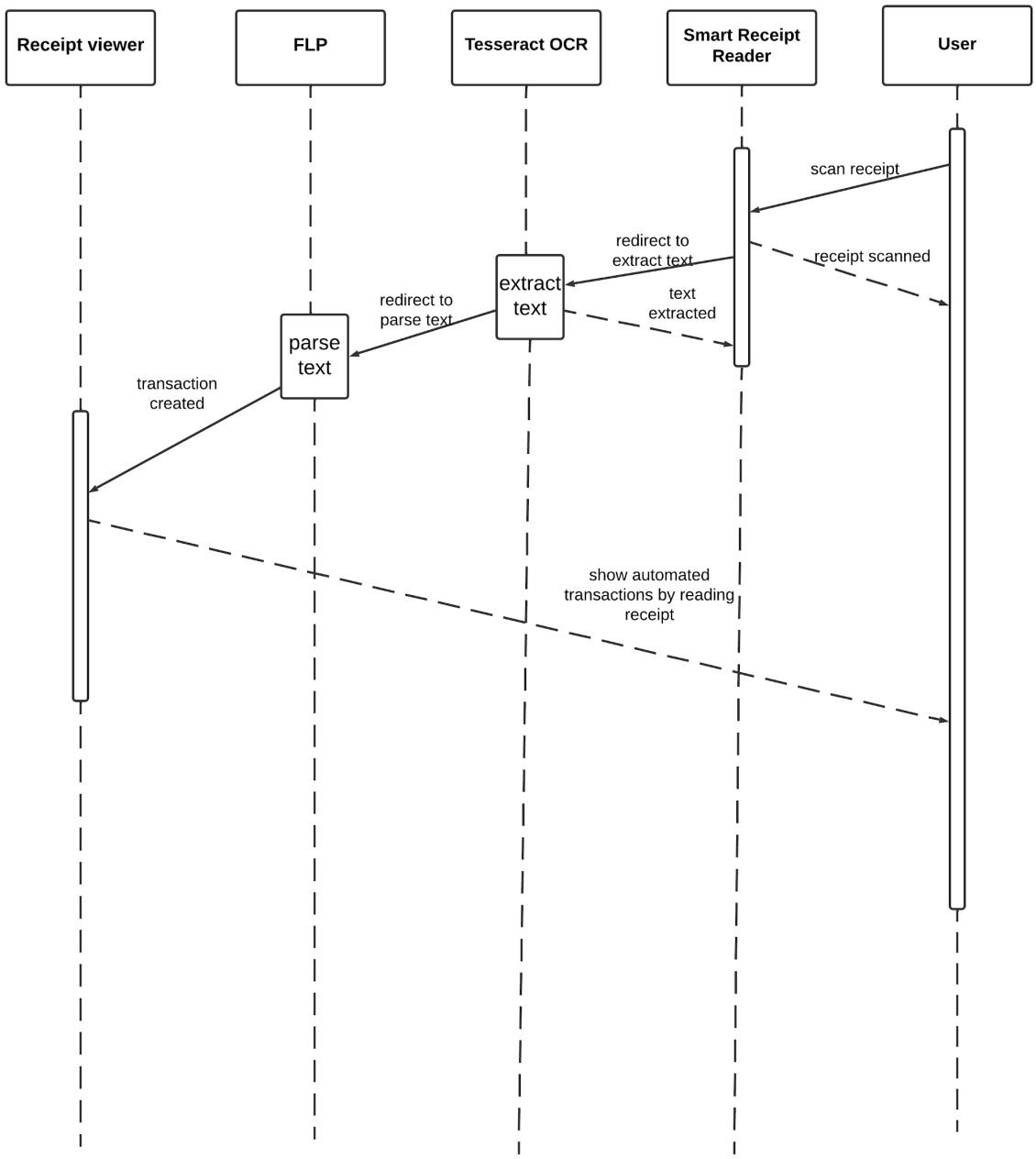

FLP
Firestore
Transaction
User
add/delete transaction
update transaction status in database
database updated
add or remove tran- -sac tions
transaction added/deleted
Generate transactions
generate transactions
verify transactions
check validity
Alternative
if generated transaction is valid
transaction is valid
verify generated transac tions
add transactions
transactions added
[Else]
transaction rejected


Firestore
Journal
User
add/delete journals
update journal
status in database
add or
rem
ove
jour
nals
database updated
journals added or
deleted
modify category of
journals
update journal
status in database
modify
journals
database updated
journal modified


Firestore
Budget
User
user want to create budget
ask to provide category, time period and currency
provide category, time period and currency
update budget status in database
create bud--get
database updated
budget added


Firestore
Goal
User
user want to create goal
ask to provide category, desired date and currency
provide category, desired date and currency
update goal status in database
create goal
database updated
goal added


Cloud Firestore
Google-Finance CDN
Currency Mode
User
select currency
update base currency
select base currency
currency updated
currency selected
select new preffered currency
retrieve exchange rate
change currency mode
provide exchange rate
preffered currency added


Receipt viewer
FLP
Tesseract OCR
Smart Receipt Reader
User
scan receipt
redirect to extract text
receipt scanned
extract text
text extracted
redirect to parse text
parse text
transaction created
show automated transactions by reading receipt